Scheduling flexible production systems under resource availability constraints

Sadia Azem

Ecole Nationale Supérieure des Mines
SAINT-ETIENNE

N° d'ordre : 2010 EMSE 0574

THÈSE

présentée par

Sadia AZEM

pour obtenir le grade de
Docteur de l'École Nationale Supérieure des Mines de Saint-Étienne

Spécialité : Génie Industriel

ORDONNANCEMENT DES SYSTEMES FLEXIBLES DE PRODUCTION SOUS CONTRAINTES DE DISPONIBILITE DES RESSOURCES

Soutenue à Gardanne, le 22 juin 2010

Membres du jury

Président :	Eric PINSON	Professeur, Université Catholique de l'Ouest, Angers
Rapporteurs :	Jean-Charles BILLAUT	Professeur, Ecole Polytechnique, Tours
	Bernard PENZ	Professeur, Institut National Polytechnique, Grenoble
Examinateur(s) :	Ammar OULAMARA	Maitre de Conférences HDR, Ecole des Mines de Nancy
Directeur(s) de thèse :	Stéphane DAUZERE-PERES	Professeur, Ecole des Mines de Saint-Etienne, Gardanne
	Riad AGGOUNE	Responsable d'Unité, Centre de Recherche Henri Tudor, Luxembourg

Spécialités doctorales :	**Responsables :**
SCIENCES ET GENIE DES MATERIAUX	J. DRIVER Directeur de recherche – Centre SMS
MECANIQUE ET INGENIERIE	A. VAUTRIN Professeur – Centre SMS
GENIE DES PROCEDES	G. THOMAS Professeur – Centre SPIN
SCIENCES DE LA TERRE	B. GUY Maître de recherche – Centre SPIN
SCIENCES ET GENIE DE L'ENVIRONNEMENT	J. BOURGOIS Professeur – Centre SITE
MATHEMATIQUES APPLIQUEES	E. TOUBOUL Ingénieur – Centre G2I
INFORMATIQUE	O. BOISSIER Professeur – Centre G2I
IMAGE, VISION, SIGNAL	JC. PINOLI Professeur – Centre CIS
GENIE INDUSTRIEL	P. BURLAT Professeur – Centre G2I
MICROELECTRONIQUE	Ph. COLLOT Professeur – Centre CMP

Enseignants-chercheurs et chercheurs autorisés à diriger des thèses de doctorat (titulaires d'un doctorat d'État ou d'une HDR)

AVRIL	Stéphane	MA	Mécanique & Ingénierie	CIS
BATTON-HUBERT	Mireille	MA	Sciences & Génie de l'Environnement	SITE
BENABEN	Patrick	PR 1	Sciences & Génie des Matériaux	CMP
BERNACHE-ASSOLANT	Didier	PR 0	Génie des Procédés	CIS
BIGOT	Jean-Pierre	MR	Génie des Procédés	SPIN
BILAL	Essaïd	DR	Sciences de la Terre	SPIN
BOISSIER	Olivier	PR 1	Informatique	G2I
BOUCHER	Xavier	MA	Génie Industriel	G2I
BOUDAREL	Marie-Reine	PR 2	Génie Industriel	DF
BOURGOIS	Jacques	PR 0	Sciences & Génie de l'Environnement	SITE
BRODHAG	Christian	DR	Sciences & Génie de l'Environnement	SITE
BURLAT	Patrick	PR 2	Génie industriel	G2I
COLLOT	Philippe	PR 1	Microélectronique	CMP
COURNIL	Michel	PR 0	Génie des Procédés	SPIN
DAUZERE-PERES	Stéphane	PR 1	Génie industriel	CMP
DARRIEULAT	Michel	IGM	Sciences & Génie des Matériaux	SMS
DECHOMETS	Roland	PR 1	Sciences & Génie de l'Environnement	SITE
DESRAYAUD	Christophe	MA	Mécanique & Ingénierie	SMS
DELAFOSSE	David	PR 1	Sciences & Génie des Matériaux	SMS
DOLGUI	Alexandre	PR 1	Génie Industriel	G2I
DRAPIER	Sylvain	PR 2	Mécanique & Ingénierie	SMS
DRIVER	Julian	DR 0	Sciences & Génie des Matériaux	SMS
FEILLET	Dominique	PR 2	Génie Industriel	CMP
FOREST	Bernard	PR 1	Sciences & Génie des Matériaux	CIS
FORMISYN	Pascal	PR 1	Sciences & Génie de l'Environnement	SITE
FORTUNIER	Roland	PR 1	Sciences & Génie des Matériaux	SMS
FRACZKIEWICZ	Anna	DR	Sciences & Génie des Matériaux	SMS
GARCIA	Daniel	MR	Génie des Procédés	SPIN
GIRARDOT	Jean-Jacques	MR	Informatique	G2I
GOEURIOT	Dominique	MR	Sciences & Génie des Matériaux	SMS
GRAILLOT	Didier	DR	Sciences & Génie de l'Environnement	SITE
GROSSEAU	Philippe	MR	Génie des Procédés	SPIN
GRUY	Frédéric	MR	Génie des Procédés	SPIN
GUY	Bernard	MR	Sciences de la Terre	SPIN
GUYONNET	René	DR	Génie des Procédés	SPIN
HERRI	Jean-Michel	PR 2	Génie des Procédés	SPIN
INAL	Karim	PR 2	Microélectronique	CMP
KLÖCKER	Helmut	DR	Sciences & Génie des Matériaux	SMS
LAFOREST	Valérie	CR	Sciences & Génie de l'Environnement	SITE
LERICHE	Rodolphe	CR CNRS	Mécanique et Ingénierie	SMS
LI	Jean-Michel	EC (CCI MP)	Microélectronique	CMP
LONDICHE	Henry	MR	Sciences & Génie de l'Environnement	SITE
MALLIARAS	George Grégory	PR 1	Microélectronique	CMP
MOLIMARD	Jérôme	MA	Mécanique et Ingénierie	SMS
MONTHEILLET	Frank	DR 1 CNRS	Sciences & Génie des Matériaux	SMS
PERIER-CAMBY	Laurent	PR 2	Génie des Procédés	SPIN
PIJOLAT	Christophe	PR 1	Génie des Procédés	SPIN
PIJOLAT	Michèle	PR 1	Génie des Procédés	SPIN
PINOLI	Jean-Charles	PR 0	Image, Vision, Signal	CIS
STOLARZ	Jacques	CR	Sciences & Génie des Matériaux	SMS
SZAFNICKI	Konrad	MR	Sciences & Génie de l'Environnement	SITE
THOMAS	Gérard	PR 0	Génie des Procédés	SPIN
TRIA	Assia		Microélectronique	CMP
VALDIVIESO	François	MA	Sciences & Génie des Matériaux	SMS
VAUTRIN	Alain	PR 0	Mécanique & Ingénierie	SMS
VIRICELLE	Jean-Paul	MR	Génie des procédés	SPIN
WOLSKI	Krzysztof	DR	Sciences & Génie des Matériaux	SMS
XIE	Xiaolan	PR 1	Génie industriel	CIS

Glossaire :

PR 0	Professeur classe exceptionnelle
PR 1	Professeur 1ère catégorie
PR 2	Professeur 2ème catégorie
MA(MDC)	Maître assistant
DR	Directeur de recherche
Ing.	Ingénieur
MR(DR2)	Maître de recherche
CR	Chargé de recherche
EC	Enseignant-chercheur
IGM	Ingénieur général des mines

Centres :

SMS	Sciences des Matériaux et des Structures
SPIN	Sciences des Processus Industriels et Naturels
SITE	Sciences Information et Technologies pour l'Environnement
G2I	Génie Industriel et Informatique
CMP	Centre de Microélectronique de Provence
CIS	Centre Ingénierie et Santé

Dernière mise à jour le : 9 mars 2010

A ceux que j'aime

Remerciements

Je tiens à témoigner ma profonde gratitude à Stéphane DAUZERE-PERES, Professeur à l'Ecole Nationale Supérieure des Mines de Saint-Etienne, et à Riad AGGOUNE, Responsable d'Unité au Centre de Recherche Henri Tudor du Luxembourg, sans qui ce travail n'aurait pas vu le jour, pour m'avoir encadrée, pour leur rigueur, leur expérience, leurs précieuses orientations et leurs conseils avisés qui m'ont guidée tout au long du parcours. Mes remerciements les plus vifs, empreints d'une grande reconnaissance à Pascal BOUVRY, Professeur à l'Université du Luxembourg, pour m'avoir si bien accueillie au sein de son équipe durant la période passée à l'Université. Je les remercie aussi pour leur soutien et les moyens mis à ma disposition.

Je remercie Jean-Charles BILLAUT, Professeur à l'Ecole Polytechnique de Tours, et Bernard PENZ, Professeur à l'Institut Polytechnique de Grenoble, d'avoir accepté d'être rapporteurs de cette thèse. Merci à vous ainsi qu'à Eric PINSON, Professeur à l'Université Catholique de l'Ouest à Angers, et Ammar OULAMARA, Maitre de Conférences HDR de l'Ecole des Mines de Nancy, pour l'intérêt et l'attention que vous avez accordé à ce travail et de m'avoir fait l'honneur de faire partie de mon Jury. Je vous témoigne ma gratitude pour ces échanges enrichissants.

Je tiens à exprimer ma grande gratitude à Christian ARTIGUES, Chargé de Recherche HDR au LAAS CNRS à Toulouse, de m'avoir proposé son aide ; et d'avoir suscité davantage mon intérêt pour la recherche pendant mon stage de master.

Ma profonde reconnaissance va également au Gouvernement du Luxembourg, plus particulièrement au Ministère de la Culture, de l'Enseignement Supérieur et de la Recherche, et au Fonds National de la Recherche, pour le financement de cette étude sans lequel cette dernière n'aurait pas pu être menée. Je les remercie chaleureusement pour leurs encouragements et leur

compréhension.

Je suis aussi reonnaissante à l'Ecole Nationale Supérieure des Mines de Saint-Etienne et l'Université du Luxembourg pour les moyens déployés pour la bonne réalisation de ce travail. Mes remerciements vont à leurs personnels que j'ai croisés ou cotoyés pour leur gentillesse et leur professionnalisme. Je témoigne, plus particulièrement, ma reconnaissance aux membres des équipes SFL et ILIAS de m'avoir offert un environnement de travail agréable et épanouissant et pour leur soutien.

Je témoigne mon ineffable reconnaissance à mes proches pour leur soutien infaillible et pour avoir fait de moi la personne heureuse que je suis. Je remercie en particulier ma famille et encore plus particulièrement mon père, ma mère , mes frères et mes soeurs qui ont été présents pour moi depuis toujours.

Pour finir, je remercie toute personne ayant contribué de loin ou de près à l'existence de ce travail.

Je garderai toujours un vif souvenir de cette merveilleuse aventure qui m'a beaucoup apporté et m'a surtout tant appris sur moi-même. J'ai essayé d'en vivre pleinement et d'en apprécier chaque moment même les plus durs.

Contents

List of Figures

List of Tables

Introduction générale - General introduction

Assurer une grande compétitivité devient une nécessité pour les entreprises ; et ce afin de faire face à la concurrence. Pour les entreprises de production ou de services, résoudre le problème d'ordonnancement de façon efficace est une expression de cette compétitivité ; il en résulte notamment la réduction des coûts et des délais.

La théorie de l'ordonnancement est une branche de la recherche opérationnelle et de la gestion de production. Ainsi, des modèles mathématiques et des méthodes de résolution sont conçues pour résoudre les problèmes posés. Un problème d'ordonnancement est défini par un ensemble de travaux à réaliser sur un ensemble de ressources ; de sorte qu'une fonction objectif soit optimisée. Ainsi, il s'agit de situer les travaux à réaliser par rapport à leur affectation aux ressources, au séquencement de leur passage sur chaque ressource, et à leur datage.

Les champs d'application de la théorie d'ordonnanacement sont diverses, notamment dans le secteur industriel tel que l'organisation de la production dans les entreprises manufacturières, celui de l'informatique tels que le partage de la mémoire, le choix des tâches à envoyer aux processeurs, et celui de l'organisation de grands projets de constructions de travaux publics tels que les chantiers routiers, ferroviaires, navales, aéronautiques.

Pour cela, les problèmes d'ordonnancement ont été largement étudiés, par des informaticiens, des automaticiens et des spécialistes de la recherche opérationnelle, depuis près d'une soixantaine d'années et plus particulièrement durant les quatres dernières décennies. Les problèmes traités par la théorie de l'ordonnancement sont classés en différentes catégories, comme les problèmes pouvant se modéliser en des problèmes d'ordonnancement d'ateliers de production (problème

à une machine, les problèmes à machines parallèles, les problèmes de type flow-shop, job-shop, open-shop) et et les problèmes d'ordonnancement de projets.

0.1 Objectif de la thèse

La majeure partie des études des problèmes d'ordonnancement se placent dans le contexte où les ressources sont disponibles en permanence. Ce qui en réalité n'est pas toujours le cas. Les différentes ressources qu'elles soient humaines ou matérielles peuvent, pour diverses raisons, être indisponibles. Les dates et les durées des indisponibilités peuvent être connues dans certains cas : congés de personnel, opérations de maintenance sur les machines etc (contexte déterministe) ; notre étude traite essentiellement ce contexte. Elles ne sont pas prévisibles dans les cas tels que des pannes de machines ou d'absences de personnel, pour des raisons médicales par exemple (contexte non déterministe) ; notre étude traite aussi ce contexte lorsqu'il est possible de refaire l'ordonnancement une fois les indisponibilités connues.

La présence de ces 'trous' dans un planning prévisionnel influe de manière significative sur le processus de production de biens ou de services et tout ordonnancement réaliste se doit d'en tenir compte. Une manière de pallier l'indisponibilité d'une ressource est d'affecter sa charge de travail à une ressource de remplacement. Mais une ressource capable d'assurer cette prise en charge, aussi bien en terme de capacité qu'en terme de compétence, n'existe pas forcément. Il est ainsi nécessaire de trouver la manière la plus appropriée de séquencer les tâches sur les ressources de façon à tenir compte de leurs indisponibilités et de l'ordre entre les tâches. Et bien qu'un plus grand effort de recherche soit déployé pour étudier les problèmes avec contraintes de disponibilité des ressources, car ils sont plus réalistes, le nombre de travaux qui leurs sont dédiés dans la littérature sur l'ordonnacement reste toujours pas considérable surtout si l'on considère des périodes d'indisponibilité flexibles.

Dans cette thèse, nous nous intéressons plus particulièrement aux problèmes de type job-shop avec des ressources non disponibles en continu, périodes d'indisponibilité flexibles et des tâches pouvant éventuellement être interrompues par les périodes d'indisponibiité. Le job-shop est l'un des problèmes de la théorie de l'ordonnancement le plus traité ; par contre sa grande complexité fait de lui l'un des problèmes de l'optimisation combinatoire les plus difficiles à résoudre ; il est donc indispensable de savoir si l'on doit privilégier la qualité de la solution

recherchée ou la rapidité du temps de calcul, ou trouver un compromis. Ceci étant aussi lié à la taille des instances traitées.

Le but de cette thèse est donc de développer des méthodes de résolution efficaces pour les problèmes d'ordonnancement rencontrés dans les systèmes flexibles de production ; et intégrant des contraintes pratiques telles que la limitation des disponibilités des ressources. Notons que l'étude de cette dernière n'est que récente malgré sa petinence au niveau industriel. En effet, cette contrainte est souvent negligée par les chercheurs ou les oblige le plus souvent à se limiter à des problèmes de tailles réduites ou à des problèmes basiques. Ceci étant du au fait que son intégation rend le problème d'ordonnancement nettement plus difficile à résoudre. Il y va sans dire qu'inclure la flexibilité sur les ressources et les tâches augmente la complexité du problème. Nous nous plaçons ainsi dans le cadre de l'étude de problèmes plus généraux, donc plus complexes, que ceux qui sont les plus présents dans la littérature. Au delà de l'apport pour les entreprises, il est évident que si l'on sait traiter les cas des systèmes flexibles, on sait de ce fait traiter les systèmes classiques. Aussi, cela devrait nous permettre d'élargir nos domaines d'investigation en y intégrant les problèmes d'ordonnancement rencontrés dans les services.

Cependant, avant de s'attaquer aux systèmes flexibles, nous nous intéressons en premier lieu à l'étude des problèmes de base avec la prise en compte de contraintes de disponibilité de ressources

Cette flexibilitité peut être relative à au moins l'un des points suivants :

- Déplacer la période d'indisponibilité dans sa fenêtre de temps; définie par sa date de début au plus tôt et sa date de fin au plus tard, permet la création d'un temps libre sur la ressource pour effectuer la tâche plus tôt.
- La durée de la période d'indisponibilité d'une ressource : En fonction d'une prise de décision managériale, une priorité peut être donnée à la production au dépend du planning d'indisponibilité des ressources.
- Une tâche peut être interrompue par une période d'indisponibilité, ensuite reprise avec une éventuelle pénalité, dès que la ressource est à nouveau disponible. Ceci peut être le cas de produits regroupés en lots.

- Une extension du problème étudié, est celui de l'ensemble de ressources pouvant effectuer une tâche donnée (cette dernière nécessite uniquement une ressource pour l'exécuter): Dans le problème classique, une opération nécessite exactement une seule ressource définie à priori pour l'effectuer. Lorsque cette ressource est non-disponible, l'opération doit attendre que la ressource redevienne disponible à nouveau. Ainsi, l'intérêt pratique de ce types d'ateliers flexibles est qu'ils permettent de modéliser de nombreux problèmes d'ordonnancement rencontrés dans le secteur des services, où il est aussi souvent nécessaire de déterminer la bonne affectation pour chaque tâche. En effet, les ressources sont souvent des personnes, et la flexibilité dans l'exécution des tâches vient d'une part du fait que plusieurs personnes de même qualification peuvent effectuer la même tâche, et d'autre part de la polyvalence des personnes qui peuvent faire différents types de tâches.

0.2 Organisation du manuscrit

Le manuscrit est organisé en deux parties:

Dans la première partie (Part I), constituée des Chapitres de 1 à 3, nous abordons l'ordonnancement de production, les techniques de résolution des problèmes d'optimisation utilisées pour l'ordonnancement, et l'état de l'art de l'ordonnancement de production avec des périodes d'indisponibilité sur les ressources.

Le Chapitre 1 décrit l'ordonnancement de production. Nous rappelons donc les structures classiques des problèmes d'odonnancement d'ateliers et nous introduisons des notations et des définitions utilisées en ordonnancement d'atelier et dans le manuscrit. Nous abordons ensuite les aspects de représentations des ordonnancements sous forme de diagramme et de graphe, et les principales notions de complexité des problèmes d'ordonnancement. Nous décrivons à la fin le contexte de l'étude et la caractérisation du problème étudié.

Dans le Chapitre 2, nous décrivons brièvement les principales méthodes, représentées par deux grandes familles, utilisées pour résoudre les problèmes d'ordonnancement. Pour chaque méthode, nous présentons l'idée générale. Seules les méthodes que nous utilisons dans notre étude sont plus détaillées. Ces deux familles de méthodes sont : les méthodes exactes qui tentent de trouver des solutions optimales à des problèmes d'optimisation combinatoire, par

une exploration intelligente de l'espace des solutions mais souvent très couteuse en terme de temps de calcul ; et les méthodes approchées, représentant une bonne alternative aux méthodes exactes, car elles permettent de trouver de bonnes solutions à moindre coût.

Le Chapitre 3 est consacré à un état de l'art couvrant les travaux de recherches menés sur les problèmes statiques et déterministes d'ordonnancement d'atelier avec indisponibilité des resssources jusqu'à 2009. Ainsi ce chapitre est dédié aux problèmes avec des périodes d'indisponibilité fixes et flexibles (on entend ici par la flexibilité la possibilité de faire varier la date de début d'une période d'indisponibilité dans un intervalle défini des dates de début au plus tôt et au plus tard). Nous remarquons que bien que des efforts de recherche ont été fournis pour l'étude de l'intégration de la contrainte de disponibilité des ressouces, ils sont essentiellement concentrés sur le cas où les périodes d'indisponibilité sont fixes. De plus, la plupart des problèmes étudiés sont : le problème à une machine, le problème à machines parallèles et le problème du flow shop. L'intérêt de ce chapitre, est de permettre de faire un état des lieux de la recherche par rapport à la problématique étudiée, et de dégager des pistes d'étude et de réflexion.

La deuxième partie (Part II), représentée par les chapitres 4, 5 et 6, présente nos principales contributions à l'étude du problème du job shop sous contraintes de disponibilité des ressources : En premier lieu une approche de modélisation mathématique, ensuite des méthodes approchées et pour finir une approche par génération de colonnes.

Mise-à-part la résolution des jeux de données dont nous disposons, le but de la modélisation mathématique, objet du Chapitre 4, est de permettre une meilleure connaissance des problèmes à travers leurs expressions mathématiques. Elle nous renseigne sur la façon de traiter au mieux les contraintes d'indisponibilité des ressources ; et de l'utilité de l'intégration de la flexibilité aux problèmes ; pas uniquement pour représenter la réalité de l'industrie mais aussi pour obtenir de meilleures solutions aux problèmes. Cette modélisation permet aussi d'évaluer la qualité des méthodes approchées et de l'aproche basée sur la génération de colonnes ; et ce compte tenu de la difficulté de trouver de bonnes bornes théoriques aux problèmes étudiés. Elle peut aussi être facilement étendue pour considérer d'autres critères d'optimisation et inclure d'autres contraintes sur les tâches. C'est une approche souvent négligée compte tenu de la complexité des problèmes concernés ; bien qu'elle peut fournir de meilleurs résultats pour un nombre

représentatif d'instances de problèmes.

Le Chapitre 5 concerne les méthodes approchées. En général, les méthodes approchées (heuristiques) représentent une alternative appropriée aux méthodes exactes pour résoudre des problèmes complexes d'optimisation combinatoire ; et ce, compte tenu de leur capacité à fournir de bonnes solutions à moindre coût. Les méthodes que nous développons pour résoudre notre problématique sont des méthodes qui permettent de construire un ordonnancement en se basant sur des règles de priorité intégrant la flexibilité des périodes d'indisponibilité et l'interruptabilité des tâches par des périodes d'indisponibilité. Nous discutons aussi, la manière dont ces méthodes, qui constituent des blocs de construction, peuvent être intégrées dans d'autre méthodes approchées pour améliorer les résultats obtenus.

Le Chapitre 6 présente une approche basée sur la génération de colonnes pour résoudre le problème du job shop avec périodes d'indisponibilité des ressources fixes et flexibles. Cette approche cherche la solution optimale par la construction de modèle commençant par un ensemble réduit de colonnes ; et à chaque fois qu'une colonne semble nécessaire pour satisfaire le problème, elle est ajoutée au modèle jusqu'à ce qu'aucune colonne n'est ajoutée. Une colonne est associée à une tâche ou à une période d'indisponibilité. Cette méthode ne peut être considérée comme étant une méthode exacte pour notre problème. Pour la rendre exacte, il aurait fallu opérer un branch and price à la fin plutôt qu'un branch and bound.

Part I

Production scheduling, optimization and state of the art

Chapter 1

Ordonnancement de production

La théorie de l'ordonnancement est une branche de la recherche opérationnelle et de la gestion de production. Un problème d'ordonnancement est défini par un ensemble de *jobs* (*tasks*) à réaliser sur un ensemble de *ressources* (*resources*) ; de sorte qu'une *fonction objectif* (*objective function*) soit optimisée.

Il s'agit, donc, de situer les jobs à réaliser par rapport à leur affectation aux ressources, au séquencement de leur passage sur chaque ressource, et à leur datage. Un ordonnancement peut être *prédictif* (*predictive*) ou *statique* (*static*) lorsqu'il s'appuie sur des données connues a priori; il peut toutefois être *réactif* (*reactive*), *dynamique* (*dynamic*) ou *temps réel* (*real time*) lorsqu'il doit s'adapter à des données intégrées a posteriori ou en temps réel.

Les problèmes d'ordonnancement ont été largement étudiés ces dernières décennies ; et ce pour la diversité de leurs champs d'application, notamment le secteur industriel (Pinedo [Pin95]) et celui de l'informatique (Blazewicz *et al* [BEPSW96]). Parmi les nombreux ouvrages de référence qui ont été publiés, on trouve Conway *et al.*[CMM67], Baker [Bak74], Rinnooy Kan [RK76], French [Fre82], Carlier et Chrétienne [CC88], Tanaev *et al.* [TGS94a] et [TGS94b], Brucker [Bru98], Esquirol et Lopez [EL99], Lopez et Roubellat [LR01], Leung [Leu04], Blazewicz *et al.* [BEPSW07].

Dans la Section 1.1 nous présentons des généralités sur les problèmes d'ordonnancement : définition d'un ordonnancement, le type de resources et les composants d'un problème d'ordonnancement. La Section 1.2 décrit les différents problèmes d'ordonnancement d'ateliers.

La Section 1.3 introduit des notations et des définitions utilisées en ordonnancement d'atelier et dans le manuscrit ; notamment les composants d'un système de production et leurs caractéristiques, la classification des problèmes d'ordonnancement et les types d'ordonnancement. Les représentations des ordonnancements sous forme de diagramme et de graphe sont presentées en Section 1.4. La complexité des problèmes d'ordonnancement est abordée dans la Section 1.5. Dans la Section 1.6 consacrée à la description du contexte de l'étude et la définition du problème etudié, les périodes d'indisponibilité des ressources et les modèles d'indisponibilité sont définis ; la caractérisation du problème est discutée ; plus particulièrement les données du problème, les contraintes auxquelles est soumis le système de production étudié et les objectifs du problème ; ainsi que l'extension du problème pour tenir compte de la flexibilité des indisponibilités des ressources.

1.1 Généralités

La terminologie utilsée en ordonnancement est issue du contexte industriel en particulier les ateliers de fabrication manufacturière.

L'ordonnancement est le processus de répartition, dans le temps, de *tâches* sur des *ressources*, et dont l'ensemble est soumis à certaines *contraintes* ; et ce afin d'optimiser un critère donné ou un compromis entre plusieurs critères exprimés par des *fonctions objectif* et qui permettent d'apprécier la qualité de tout séquencement de ces tâches appelé *ordonnancement* (*schedule*). Ces critères à optimiser peuvent être liés à l'utilisation des ressources telles que la charge des machines d'un atelier, ou au temps comme la date d'achèvement de la réalisation des jobs (*makespan*). Le critère est dit *régulier* lorsque l'avancement de l'exécution d'une tâche, sans en retarder d'autres, ne dégrade pas la valeur du critère; autrement dit s'il est fonction décroissante des dates d'achèvement des opérations.

Lorsque la quantité d'une ressource diminue au fur et à mesure de son utilisation (matières premières, financement d'un projet, ...), elle est dite *consommable* (*consumable*). Lorsqu'elle demeure disponible en même quantité (équipe, machine d'un atelier), elle est dite *renouvelable* (*renewable*) ou non-partageable (non-shared). Lorsqu'elle ne peut exécuter qu'une opération à la fois, elle est dite *disjonctive* (*disjunctive*) ; autrement elle est dite *cumulative*.

Dans ce qui suit nous utiliserons la terminologie job, machine (au lieu de tâche et ressource).

Le passage d'un job sur une machine est appelé *opération* (*operation*). Cette dernière possède des caractéristiques temporelles : sa *durée opératoire* (*processing time*) (durée d'exécution), sa *date de disponibilité* (*release date*) (date à laquelle elle peut être exécutée au plus tôt), qui lorsqu'elle existe est impérative, sa *date de fin au plus tard souhaitée* (*due date*) ou *date d'échéance*, les *contraintes de précédence* (*precedence constraints*), et qui représente un ordre partiel des opérations. La date d'échéance peut souvent être violée au prix de pénalités diverses ; si elle est impérative, elle est appelée *deadline*. L'opération est soit *non-préemptive* (*non-preemptive*) si elle doit être réalisée sans interruption, soit *préemptive* (*preemptive*) si elle peut être effectuée par morceaux.

1.2 Problèmes d'ordonnancement d'atelier

Les problèmes sont caractérisés par le nombre de machines dans l'atelier et leur disposition, des nombres d'opérations composant les jobs, et des ordres de leurs passages sur les machines ; ainsi que le nombre de machines pouvant réaliser une opération.

Dans les problèmes d'ordonnancement d'atelier, les ressources sont disjonctives (Esquirol et Lopez, [EL99]) ou de capacité unitaire et sont indépendantes les unes des autres. Il en est de même pour les jobs.

On distingue donc les problèmes à une machine et les problèmes multi-machines (machines parallèles, flow shop, flow shop hybride, job shop, job shop flexible et open shop).

1.2.1 Problème à une machine

Le problème d'atelier à une machine (single machine problem) consiste à ordonnancer, sur une seule machine, des jobs constitués d'une seule opération. Pour la minimisation du makespan (dure totale de l'ordonnancement) pour des jobs disponibles à l'instant 0, toute séquence d'exécution aboutit à une solution optimale. Cependant, l'ajout de contraintes et la considération d'autres critères rendent le problème plus difficile à résoudre. (Voir Figure 1.1)

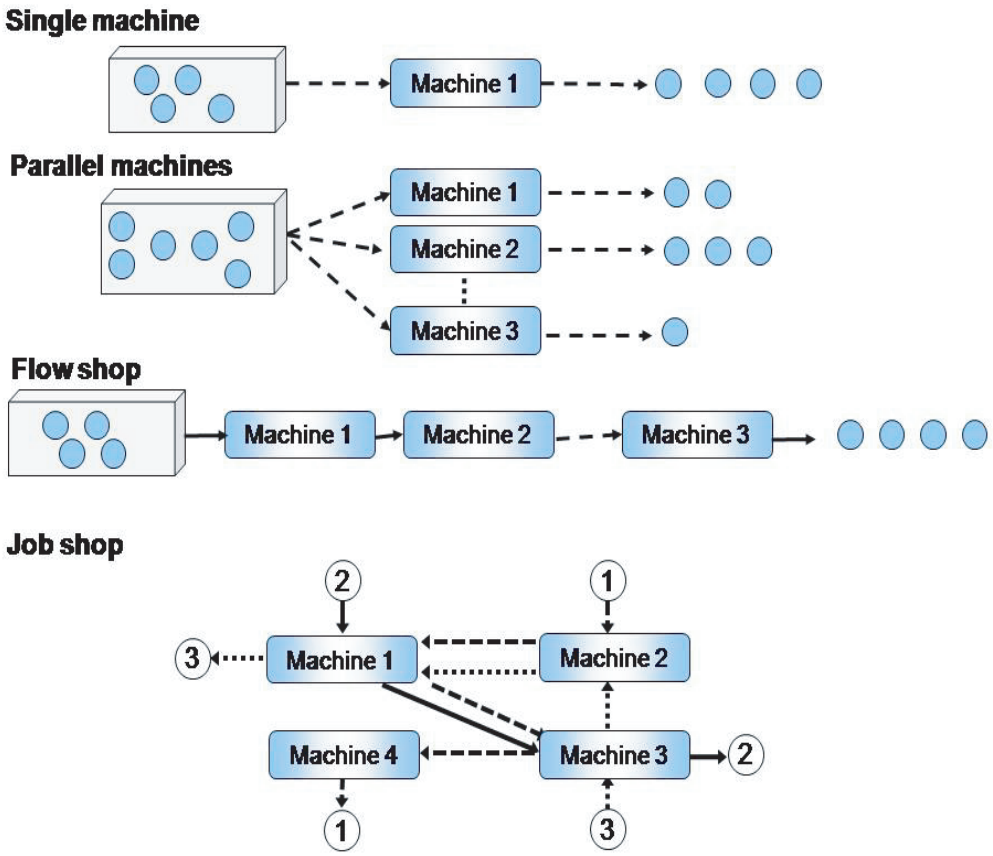

Figure 1.1: Exemples de problèmes d'atelier.

Le fait que, dans un atelier, une machine peut constituer un goulet d'étranglement rend l'étude de ce probleme intéressante ; car sa résolution permet d'aborder des problèmes plus complexes.

1.2.2 Problème multi-machines à machines parallèles

Le problme d'ordonnancement à machines parallèles (parallel machines scheduling problem) est une généralisation du problème d'atelier à une machine et un cas particulier de problème d'atelier multi-machines. Chaque job est constitué d'une seule opération et chaque opération peut être réalisée par n'importe laquelle des machines, disposées en parallèle ; mais n'en nécessite qu'une seule. Le problème revient donc à déterminer l'affectation des opérations aux machines ; ainsi que leurs dates d'exécutions. (Voir figure 1.1)

Il en existe trois types dans la littérature :

- Les problèmes à machines identiques : les durées opératoires sont égales et ne dépendent donc pas des machines,
- Les problèmes à machines uniformes : la durée d'une opération varie uniformément en fonction de la performance de la machine choisie,
- Les problèmes à machines indépendantes (non liées) : les durées opératoires dépendent complètement des machines utilisées.

1.2.3 Problèmes d'atelier multi-machines à cheminement unique (Flow shop)

Problème de type Flow shop

Ici les machines sont disposées en série ; et les jobs à réaliser sont composés de plusieurs opérations et visitent toutes les machines selon une *gamme opératoire* (*job routing*) (*gamme de fabrication*) unique (dans le même ordre ou flôt unidirectionnel). Cette dernière est une donnée du problème (Voir figure 1.1). Lorsque le séquencement des jobs est le même sur toutes les machines, le problème est celui du flow shop *de permutation*.

Problème du Flow shop hybride (flexible)

C'est une généralisation des environnements du flow shop et des machines parallèles. L'atelier est organisé en étages constitués d'un ensemble de machines en parallèle. Cependant, une opération n'en nécessite qu'une seule pour son exécution. Les différents jobs à réaliser doivent passer sur tous les étages dans le même ordre. Ceci revient donc à trouver pour chaque job la machine exécutant l'opération associée à chaque étage, ainsi que les dates d'exécution des différentes opérations.

1.2.4 Problèmes d'atelier multi-machines à cheminements multiples (Job shop)

Problème de type Job shop

C'est une généralisation de celui du flow shop. En effet, l'ordre de passage des jobs sur les machines peut être différent d'un job à l'autre (flôt multi- directionnel ; voir figure 1.1). Lorsqu'un job peut passer sur une machine plus d'une fois, la gamme est dite *bouclante* (*recirculation*).

Problème du Job shop flexible

C'est une extension du problème classique du job shop. La différence est que pour le problème du job shop flexible, chaque opération peut être effectuée par une seule machine dans un ensemble de machines. Le problème est ainsi de déterminer à la fois une affectation et un séquencement des opérations sur les machines en fonction de l'objectif à atteindre.

Les approches hiérarchiques (*hierarchical approaches*) résolvent le problème d'abord par l'affectation des opérations aux machines et ensuite du séquencement sur chaque machine. Les approches intégrées (*integrated approaches*) résolvent simultanément les problèmes d'affectation et de séquencement.

Le problème du job shop flexible est pertinent dans au moins deux types de systèmes de production. Dans les systèmes manufacturiers flexibles, les machines peuvent effectuer différents types d'opérations. Le deuxième type consiste en des ateliers avec des pools de machines parallèles, où les machines d'un pool peuvent effectuer uniquement un seul type d'opérations (machines dédiées). Le second type de systèmes peut être considéré comme un cas particulier du premier.

1.2.5 Problèmes d'atelier multi-machines à cheminements libres (Open shop)

Contrairement au problème du job shop, dans l'open shop, les gammes opératoires des différents jobs ne sont pas fixées a priori. Les opérations d'un même job peuvent donc être exécutées dans un ordre quelconque. Le problème consiste d'une part à déterminer le cheminement de chaque job et d'autre part à ordonnancer les jobs en tenant compte des gammes trouvées. Ces deux problèmes peuvent être résolus simultanément. Comparé aux autres modèles d'ateliers multi-machines, l'open-shop n'est pas très étudié dans la littérature; ceci étant dû au fait qu'il n'est pas courant dans les entreprises.

1.3 Définitions et notations

1.3.1 Notations

n : nombre de jobs,
$J = \{J_1, J_2, \cdot\cdot, J_n\}$: ensemble des jobs à réaliser,

n_i : nombre d'opérations du job J_i,

r_i : date de disponibilité (*release date* or *ready date*) du job J_i,

d_i : date de fin souhaitée (*due date*) de J_i,

$\tilde{d}_i$: date de fin impérative (*deadline*) de J_i,

w_i : coefficient de pondération (*weight*) associé à J_i,

C_i : date de fin (*completion date*) de J_i,

L_i : écart par rapport à la fin souhaitée ou retard algébrique (*lateness*) du job J_i . $L_i = C_i - d_i$,

E_i : avance (*earliness*) du job J_i. $E_i = max(d_i - C_i, 0)$,

T_i : retard (*tardiness*) du job J_i. $T_i = max(C_i - d_i, 0)$,

U_i : indicateur de retard (*unit penalty*) du job J_i . $U_i = 1$ si $T_i > 0$, $U_i = 0$ sinon,

O_{ij} : j^{eme} opération du job J_i,

t_{ij} : date de début (starting date) de O_{ij},

p_{ij} : durée opératoire (*processing time*) de O_{ij},

C_{ij} : date de fin (*completion date*) de O_{ij},

m : nombre de machines,

$M = \{M_1, M_2, \cdot\cdot, M_m\}$: ensemble des machines de l'atelier.

1.3.2 Classification des problèmes d'ordonnancement

La notation la plus utilisée en ordonnancement, introduite par Graham *et al.* [GLLRK79], décrit les problèmes d'ordonnancement en trois champs $\alpha|\beta|\gamma$:

- $\alpha = \alpha_1\alpha_2$: environnement des machines.
  - α_1 : représente le type d'atelier. Il peut prendre les valeurs 1, P, Q, R, F, FF, J, FJ, O qui correspondent respectivement aux problèmes à une seule machine, à machines parallèles identiques, à machines parallèles uniformes, à machines parallèles non liées, de type flow shop, de flow shop flexible, de type job shop, de job shop flexible et de type open shop.
  - α_2 : représente le nombre de machines.
  - D'autres paramètres peuvent être rajoutés au champ α exemple h_k (resp. h_{rk}) représente les k périodes d'indisponibilités dans un problème à une machine (resp. sur la machine M_r).

- β : Ensemble des contraintes sur les jobs. On trouve, par exemple, :
  - *pmtn* : la préemption est autorisée,
  - *ppmtn* : préemption partielle.
  - *prec* : existence de contraintes générales de précedence entre les opérations,
  - $p_{ij} = p$: toutes les durées opératoires sont égales à p,
  - r_i : chaque job J_i possède une date de disponibilité,
  - d_i : chaque job J_i possède une date échue,
  - $S_{ii'}$: temps de préparation dépendant de la séquence entre les jobs J_i et $J_{i'}$,
  - S_{ri} : temps de préparation de la machine M_r pour le job J_i,
  - M_r^k : la machine M_r possède k périodes d'indisponibilité,
  - M_i : restriction d'admissibilité des machines (machine eligibility restriction). L'ensemble M_i désigne l'ensemble de machines pouvant effectuer le job J_i,
  - *prmu*: l'ordre (ou permutation) selon lequel les jobs passent sur la première machine est maintenu à travers le système,
  - *block* : le job complété doit rester sur la machine en amont prevenant ou bloquant cette machine d'effectuer un autre job,
  - *nwt* : no-wait implique que les jobs ne peuvent attendre entre deux machines successives,
  - *recrc* : recirculation implique qu'un job peut passer sur une machine ou un pool plus d'une fois,
- γ : représente la fonction objectif à optimiser
  - $C_{max} = max\{C_i, i = 1, \cdot\cdot, n\}$: date de fin de tous les jobs ou *makespan*. Il correspond à la date de fin de la dernière opération de l'ordonnancement. Un makespan minimum implique usuellement une haute utilisation des machines (productivité),
  - $\sum_i C_i$: somme des dates de fin des opérations. On le réfère aussi comme *flow time*. Ainsi, la somme pondérée des dates de fin $\sum_i w_i C_i$ est désignée comme le flow time pondéré. Cela donne une indication sur le coût d'exploitation et d'inventaire induits par l'ordonnancement (minimisation des encours),

- $L_{max} = max\{L_i, i = 1, \cdot\cdot, n\}$: retard algébrique maximum. Il mesure la pire violation des dates échues,
- $T_{max} = max\{T_i, i = 1, \cdot\cdot, n\}$: retard maximum,
- $\sum_i T_i$: somme des retards sur les dates d'achèvement des jobs. $\sum_i w_i T_i$: somme pondéré des retards,
- $E_{max} = max\{E_i, i = 1, \cdot\cdot, n\}$: maximum des avances,
- $\sum_i U_i$: nombre de jobs en retard. $\sum_i w_i U_i$: nombre pondéré de jobs en retard,

A titre d'exemple, $J_m|pmtn|C_{max}$ désigne le problème de la minimisation du makespan C_{max} dans un atelier de type job shop à m machines et où la préemption est autorisée.

1.3.3 Types d'ordonnancement

Il existe différents types d'ordonnancement définis comme suit : Un ordonnancement est *semi-actif* (*semi-active*) lorsqu'il est impossible d'avancer une opération sans modifier la séquence des opérations sur la ressource. Il est dit *actif* (*active*) s'il est impossible d'avancer une opération sans reporter le début d'une autre opération. Il est dit *sans retard* ou *sans délai* (*non-delay*) si et seulement si aucune opération n'est mise en attente lorsqu'une machine est disponible pour l'exécuter. Ainsi, les ordonnancements sans retard sont inclus dans le sous-ensemble des ordonnancements actifs ; qui sont eux mêmes inclus dans le sous-ensemble des ordonnancements semi-actifs. Baker [Bak74] a énoncé la propriété suivante : l'ensemble des ordonnancements semi-actifs est *dominant* dans les problèmes d'optimisation d'un critère régulier et le sous-ensemble des ordonnancements actifs est le plus petit ensemble dominant.

1.4 Représentations des ordonnancements

Les solutions d'un problème d'ordonnancement d'atelier peuvent être représentées par le diagramme de Gantt et le graphe disjonctif.

1.4.1 Exemple 1.1

Soit le problème $J_3|n = 4|C_{max}$ dont les gammes opératoires sont les suivantes :

J_1 : M_1 (1) M_2 (2) M_3 (3)

J_2 : M_2 (1) M_1 (2) M_3 (3)

J_3 : M_3 (2) M_2 (1) M_1 (3)

J_4 : M_1 (4) M_3 (1) M_2 (1)

Les nombres mis entre parenthèses représentent les durées opératoires.

1.4.2 Diagramme de Gantt

Le diagramme de Gantt (par H. Gantt) permet de montrer les séquences de traitement sur chaque machine et les dates de début et de fin des jobs. En effet, il se compose de lignes horizontales désignant les machines ; les opérations y sont représentées, en fonction des machines correspondantes, à partir de leur dates de début d'exécution, sous forme de barres ayant des longueurs proportionnelles à leur durées opératoires.

A titre d'illustration une solution réalisable du problème de l'exemple 1.1 est donnée par la Figure (1.2).

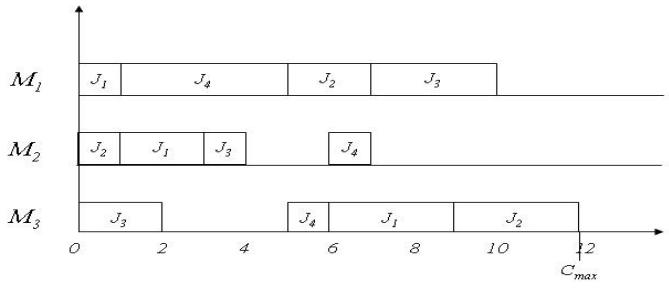

Figure 1.2: Diagramme de Gantt associé à l'exemple 1.1.

1.4.3 Graphe disjonctif

Le graphe disjonctif, fut proposé par Roy et Susmann [RS64] pour le problème du job shop. Soit $G = (N, A_1 \bigcup A_2)$ ce graphe. L'ensemble N, désignant les sommets, est constitué des opérations des jobs ainsi qu'une source (représentant le début de tous les jobs) et un puits (représentant leur fin) fictifs. L'ensemble A_1 désigne les arcs conjonctifs reliant chaque paire d'opérations consécutives d'un même job, la source (resp. le puits) à la première (resp. dernière) opération de chaque job. L'ensemble A_2 désigne les arcs disjonctifs, reliant deux opérations de jobs distincts qui utilisent une même machine. Chaque arc disjonctif consiste en une paire d'arcs d'orientations opposées tel que chaque chemin du graphe peut contenir au plus l'un d'entre eux.

Une orientation de chaque arc disjonctif (de sorte à avoir un graphe acyclique, ie. sans circuit) permet d'obtenir une solution réalisable du problème d'ordonnancement. Le graphe disjonctif est dit alors *arbitré* (*arbitrated*) (Carlier et Chrétienne [CC88]). L'exemple précédent est illustré par la Figure (1.3). La solution représentée est la même que celle du diagramme de Gantt de la Figure (1.2). Notons que la valeur du makespan est donnée par la longueur du plus long chemin allant de la source au puits.

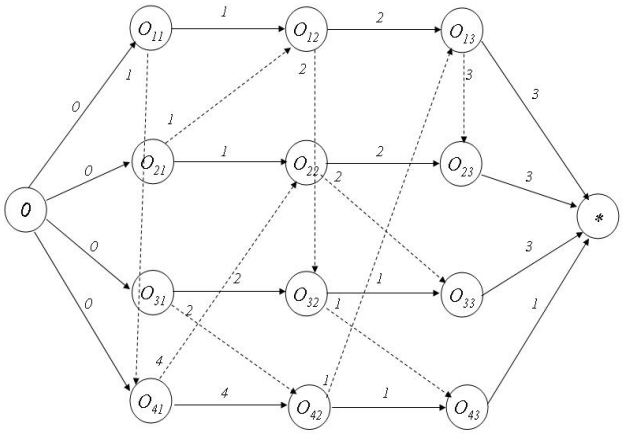

Figure 1.3: Graphe disjonctif arbitré associé à l'exemple 1.1..

1.5 Complexité des problèmes d'ordonnancement

La complexité des algorithmes est un indicateur de performance permettant de comparer les solutions à un problème. Elle représente le dénombrement des opérations élémentaires (affectation, comparaison, opérations arithmétiques, évaluation d'une expression, ...) effectuées par l'algorithme. Ainsi, si l'on dispose d'un algorithme polynômial (en fonction de la taille des données), exemple $O(n^2), O(n^3), ..$, cela signifie que le nombre d'opérations élémentaires est majoré par $c \times n^2$, $c \times n^3$,... où c est une constante.

La théorie de la complexité ([Coo71], [Kar72], [GJ79]) permet d'analyser les coûts de résolution des problèmes d'optimisation combinatoire et de classifier les problèmes en plusieurs classes de difficulté. Pour un problème donné, la distinction est faite entre problème d'optimisation et *problème de décision* (*decision problem* ou de reconnaissance et dans lequel la réponse attendue à une question donnée est oui ou non) associé en transformant par exemple la fonction objectif en une fonction binaire.

Les problèmes de classe P dits *polynômiaux* (*polynomial*) sont des problèmes pour lesquels il existe des algorithmes qui les résolvent en un temps polynomial des tailles des problèmes. La classe des problèmes NP (NP pour *Non deterministic Polynomial*) est la classe des problèmes de décision pouvant être résolus par un algorithme polynômial non déterministe. La *classe NP-complet* (*NP-complete class*) est une sous-classe des problèmes NP. Un problème est NP-complet quand tous les problèmes appartenant à NP lui sont réductibles. Ainsi, si on trouve un algorithme polynômial pour un problème NP-complet, on trouve automatiquement une résolution polynômiale de tous les problèmes de classe NP. Un problème d'optimisation est dit *NP-difficile* (*NP-hard*) si le problème de décision qui lui correspond est NP-complet. Si on peut construire des algorithmes appelés algorithmes *pseudo-polynômiaux* (*pseudo-polynomial*), qui sont des algorithmes polynômiaux en fonction de la longueur des données (taille mémoire) ; le problème étudié est alors dit NP-complet au *sens faible* (*in a weak sense*) ; autrement, il est NP-complet au *sens fort* (*in a strong sense*).

1.6 Contexte de l'étude et définition du problème étudié

1.6.1 Périodes d'indisponibilité des machines et modèles d'indisponibilités

Dans la plupart des travaux dédiés à l'ordonnancement de production, les machines sont supposées être disponibles en continu pour effectuer des jobs. Ceci n'est pas toujours vrai : les différentes ressources aussi bien matérielles qu'humaines peuvent être indisponibles pour diverses raisons. Les dates et les durées des périodes d'indisponibilité sont connues dans certains cas : congés du personnel, activités de maintenance des machines, etc. D'autres périodes d'indisponibilité telles que les pannes machines ou les défections du personnel ne sont pas prévisibles.

La présence des "trous" dans un planning influence les processus de production de façon significative. Le problème étant qu'une ressource additionnelle pour absorber cette charge de travail n'est pas forcément disponible. Il est alors nécessaire de trouver la meilleure façon de répartir la charge de travail entre les machines en prenant en compte ces périodes d'indisponibilité, le type des opérations que les machines peuvent effectuer, l'ordre entre les opérations.

Il existe dans la littérature quatre cas de figure pour lesquels une opération peut être interrompue par une période d'indisponibilité ; et qui sont illustrés par la Figure 1.4 : opérations strictement non-préemptives (*non-preemptive*), sécables (*resumable*), non-sécables (*non-resumable*), semi-sécables (*semi-resumable*). Le premier cas est dû à Aggoune [Agg02,Agg04] et les trois autres à Lee [Lee96,Lee97,Lee99]. Une opération est dite *strictement non-préemptive* lorsqu'elle ne peut être interrompue ni par une autre opération ni par une période d'indisponibilité. Une opération interrompue par une période d'indisponibilité est dite *sécable* si son exécution peut continuer aussitôt que la machine qui l'exécute est de nouveau disponible. Elle est dite *non-sécable* si elle doit recommencer complètement. Il est important de noter que ce cas est différent du cas non-préemptif du fait que dans ce dernier, lorsque l'opération ne peut être effectuée avant la période d'indisponibilité, elle doit commencer et se terminer après. Une opération est dite *semi-sécable* si elle doit partiellement recommencer lorsque la machine est de nouveau disponible. Noter que l'étude du cas semi-sécable inclue les cas sécable et non-sécable. Cependant, l'étude du cas non-sécable est moins pertinente à considérer que les autres cas ; car nous

étudions des périodes d'indisponibilité prévues.

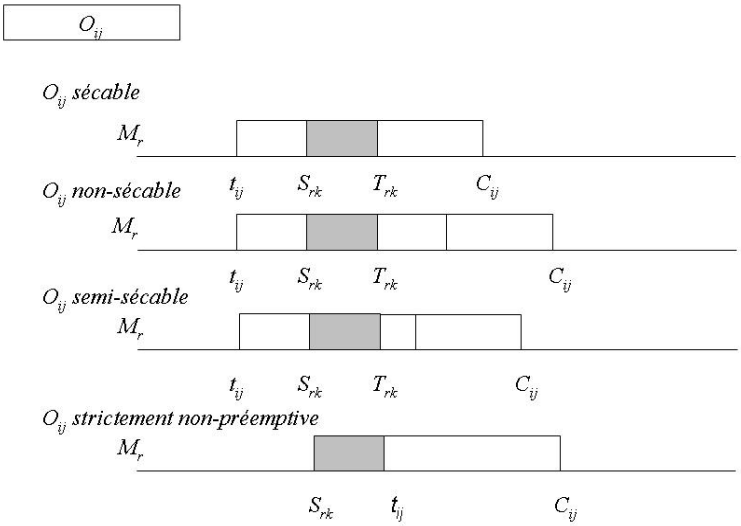

Figure 1.4: Les différents cas d'interruption d'une opération

Il existe une autre terminologie introduite dans Mauguière *et al.* [MBB05]. Elle concerne les périodes d'indisponibilité permettant l'interruption d'opérations : périodes d'indisponibilité traversable (crossable) et non-traversable (non-crossable) et qui sont représentées par la Fi- gure 1.5. Ainsi, une période d'indisponibilité est dite *traversable* si elle permet l'interruption d'une opération ; bien entendu, cette interruption ne se fera pas si l'opération est non-préemptive. Une période d'indisponibilité qui ne permet pas l'interruption d'une opération est dite *non-traversable* ; dans ce cas, l'opération ne sera pas interrompue même si elle est préemptive. Mauguière *et al.* [MBB05] énumérent tous les cas générés par les caractères sécable et traversable des opérations et des périodes d'indisponibilité et les classifient pour définir de nouveaux problèmes. Les cas sécable et non-sécable y sont étudiés.

Lorsque les machines ne sont pas disponibles en continu, le champ α contient au moins l'un des éléments suivants :

- *cr* : si les périodes d'indisponibilité sont traversables,

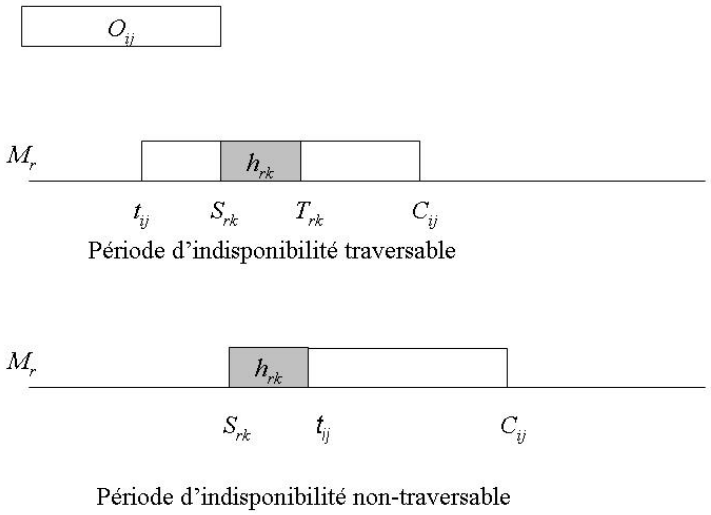

Figure 1.5: Période d'indisponibilité permettant l'interruption d'une opération

- *ncr* : si les périodes d'indisponibilité sont non-traversables,
- *brkdwn* : panne machines (machine breakdown) implique que les machines ne sont pas disponibles en continu, et représente le contexte non-déterministe.

et le champ β contient au moins :

- rs : si les opérations sont sécables,
- nrs : si les opérations sont non-sécables,
- srs : si les opérations sont semi-sécables,

Pour modéliser le problème d'indisponibilité des machines par le graphe disjonctif $G = (N, A_1 \bigcup A_2)$, présenté dans la Section 1.4.3, Aggoune [Agg02, Agg04] a introduit l'idée suivante : Chaque machine peut être considérée comme un job dont les opérations sont les périodes d'indisponibilité. N contient donc aussi les périodes d'indisponibilité des machines. A_1 contient

aussi les arcs qui relient deux périodes d'indisponibilité consécutives sur une machine, et la source (resp. le puits) et la première (resp. la dernière) période d'indisponibilité. A_2 contient aussi les arcs qui lient les opérations et les périodes d'indisponibilité sur la même machine.

1.6.2 Caractérisation du problème

Les données du problème sont :

- Un ensemble de n jobs $J = \{J_1, J_2, \cdot\cdot, J_n\}$ doit être réalisé par un ensemble de m machines $M = \{M_1, M_2, \cdot\cdot, M_m\}$,
- Chaque job J_i est composé d'une gamme opératoire qui est une séquence linéaire de n_i opérations $\{O_{i1}, O_{i2}, \cdot\cdot, O_{ij}, \cdot\cdot, O_{in_i}\}$. Cette séquence ne dépend que du job et peut varier d'un job à l'autre,
- L'opération O_{ij} est la j^{eme} opération de la gamme opératoire de J_i. Les dates de début et de fin calculées sont désignées par t_{ij} et C_{ij},
- L'opération O_{ij} nécessite l'utilisation de la machine mr_{ij} pendant p_{ij} unités de temps appelé durée opératoire connue à l'avance (aucune incertitude sur sa détermination),
- Chaque machine M_r possède m_r périodes d'indisponibilités $\{h_{r1}, h_{r2}, \cdot\cdot, h_{rk}, \cdot\cdot, h_{rm_r}\}$,
- La période d'indisponibilité h_{rk} est la k^{eme} période d'indisponibilité de M_r. Lorsque sa date de début S_{rk} n'est pas connue à l'avance, une fenêtre de temps $[ES_{rk}, LS_{rk}]$ est définie pour S_{rk} où ES_{rk} (resp. LS_{rk}) est sa date de début au plus tôt (resp. au plus tard). De même, sa durée p'_{rk}, si elle n'est pas connue à l'avance, peut être choisi dans un ensemble de valeurs,
- Nous associons pour chaque opération O_{ij} un *coefficient de pénalité sur la premption* (*coefficient of penalty on preemption*) α_{ij} qui représente la partie de l'opération O_{ij} à refaire après la période d'indisponibilité l'interrompant ; il représente son caractère sécable. Ainsi, $\alpha_{ij} = 0$ (resp. $\alpha_{ij} = 1$) si O_{ij} est sécable (resp. non-sécable) et $0 \leq \alpha_{ij} \leq 1$ si O_{ij} est semi-sécable. Pour le cas semi-sécable, Lee [Lee99] étudie dans son article le problème du flow shop à 2-machines où chaque machine a exactement une période d'indisponibilité. Pour une opération O_{ij} s'effectuant sur la machine M_r et interrompue par la période d'indisponibilité h_{r1}, il introduit un coefficient $\alpha \in [0, 1]$ qui, multiplié par la proportion de l'opération effectuée avant le début de la période d'indisponibilité h_{r1},

donne la proportion à refaire après la fin de h_{r1}. Ce coefficient est le même pour toutes les opérations,

- Nous associons un *coefficient de préemption* (*coefficient of preemption*) β_{ijk} à une opération O_{ij} qui doit s'effectuer sur la machine M_r et qui peut être interrompue par une période d'indisponibilité h_{rk}. Il représente le caractère préemptif de O_{ij} ou le caractère traversable de h_{rk}. Ainsi, $\beta_{ijk} = 0$ si h_{rk} est non-traversable ou O_{ij} est non-préemptive. Et $\beta_{ijk} = 1$ si h_{rk} est traversable et O_{ij} est préemptive. Dans ce cas, la position de l'opération O_{ij} par rapport à la période d'indisponibilité h_{rk}, dépend de son caractère sécable.

Les contraintes du problème représentent les contraintes technologiques auxquelles sont soumis les jobs et les machines. Elles concernent l'utilisation des machines et les liens existant entre les opérations.

- Les jobs sont indépendants les uns des autres,
- Il existe aussi une indépendance entre les machines,
- Chaque machine ne peut réaliser qu'une opération à la fois,
- Chaque opération ne nécessite qu'une machine à la fois. Notre étude concernant essentiellement le problème du job shop, cette machine est fixée a priori. Toutefois, dans certains cas, des extensions de l'étude sont données pour le job shop flexible,
- Aucune préemption entre les opérations n'est autorisée ; i.e., lorsque deux opérations doivent s'effectuer sur la machine, aucune d'entre elles ne doit interrompre l'autre,
- La préemption entre une opération et une période d'indisponibilité est autorisée.

Les deux critères objectifs considérés dans cette étude sont : la minimisation du makespan C_{max} et de la somme des dates de fins des jobs $\sum_{i=1}^{n} C_i$. Ainsi, l'objectif est de déterminer les séquences d'entrée des opérations sur les machines de sorte à minimiser l'un de ces critères.

Ainsi, il faudrait déterminer la date de début t_{ij} de chaque opération O_{ij} ; ainsi que sa date de fin C_{ij} dans le cas où l'interruption de O_{ij} par une période d'indisponibilité est autorisée.

Le problème du job shop avec périodes d'indisponibilité est NP-difficile au sens fort. En effet, Blazewicz *et al.* [BBFKS01] et Kubiak *et al.* [KBFBS02] démontrent que le problème du

flow shop à 2-machines est fortement NP-difficile. Nous rappelons que le flow shop est un cas particulier du job shop où les gammes opératoires sont identiques.

1.6.3 Extension à la flexibilité des indisponibilités des ressources

L'introduction des contraintes d'indisponibilité des ressources rend les problèmes classiques plus réalistes. Bien que de plus en plus d'études sont dédiées à ce type de problème, leur nombre reste faible en comparaison avec les problèmes sans périodes d'indisponibilité. L'une des raisons est que l'intégration de ces contraintes augmente la complexité du problème.

Il apparait aussi que s'interesser à des systèmes flexibles est plus réaliste ; mais augmente la complexité des problèmes associés. Cette flexibilité peut être liée à au moins un des points suivants :

- La date de début de la période d'indisponibilité machine, introduite par Aggoune [Agg02, Agg04] : déplacer la période d'indisponibilité dans sa fenêtre de temps (définie par des dates de début au plus tôt et au plus tard) permet la création d'un temps libre sur la machine pour effectuer une opération soit avant la période d'indisponibilité soit après ; ce qui permet à l'opération de finir plus tôt tel que le montre la Figure 1.6.
- La durée de la période d'indisponibilité machine : en fonction d'une décision de management, une priorité peut être donnée à la production. En effet, il est possible en élaborant un ordonnancement de modifer les durées des périodes d'indisponibilité pour chaque machine pour qu'elle ait un nombre minimum de périodes d'indisponibilité et qu'elle respecte une durée globale minimum d'indisponibilté.
- L'interruption (préemption) d'une opération par une période d'indisponibilité avec ou sans pénalité : lorsque la préemption est autorisée, une opération peut être interrompue par une période d'indisponibilité, ensuite reprise avec une éventuelle pénalité, dès que la machine est à nouveau disponible. Ceci peut être le cas de produits regroupés en lots.
- Une extension du problème étudié, est celui de l'ensemble de ressources pouvant effectuer une opération donnée (cette dernière nécessite uniquement une ressource pour l'exécuter) : Dans le problème classique, une opération nécessite exactement une seule ressource définie a priori pour l'effectuer. Lorsque cette ressource est non-disponible, l'opération doit attendre que la ressource redevienne disponible à nouveau. Ce qui n'est pas le cas si plus d'une machine peut exécuter l'opération.

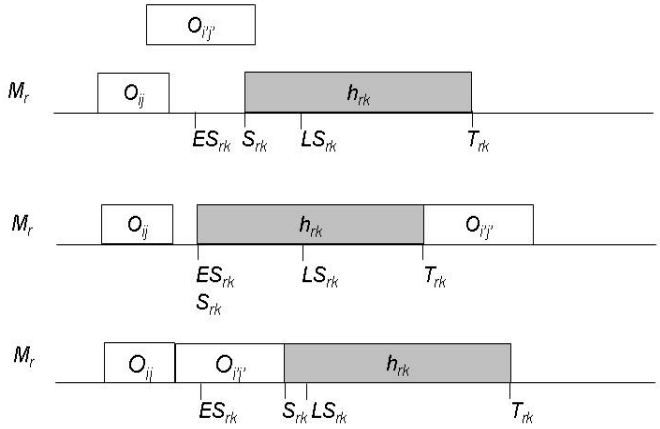

Figure 1.6: Fenêtre de temps pour une date de début d'une période d'indisponibilité

1.7 Conclusion

Dans ce chapitre, nous avons abordé tout d'abord l'ordonnancement de manière générale et de manière plus particulière l'ordonnancement de production ; et ce pour situer à la fin le contexte de notre étude. Ainsi, nous avons présenté des notations, des définitions, la classification des problèmes d'ordonnancement, les types et les représentations des ordonnancements et la complexité des problèmes d'optimisation. Pour délimiter les contours de notre étude, nous avons défini les indisponibilités des ressources et leurs flexibilités, les données, contraintes, objectifs et complexité du problème étudié.

Chapter 2

Optimization techniques

This chapter describes the main methods that are used to solve scheduling problems. For each method, the general idea is reported. More details are only given for methods we use in our study. Most of the given references concern the solution of production scheduling problem under resource availability constraints.

Section 2.1 concerns exact methods that try to find optimal solutions to optimization problems. Section 2.2 presents the approximation methods that are used as an alternative to exact methods for finding good solutions.

2.1 Exact methods

Exact methods can find optimal solutions for combinatorial optimization problems, thanks to an intelligent exploration of the solutions space, but not systematically in a polynomial time. The most commonly used methods are described in the following sub-sections:

2.1.1 Branch-and-bound procedure

Branch-and-bound procedure, introduced by Dantzig *et al.* [DFJ54] for the resolution of the traveling salesman problem, is a search method by implicit enumeration of solutions that corresponds to a tree construction avoiding unnecessary branches; and which root corresponds to the solutions space of the original problem. In a minimization problem, an upper bound UB of the objective function is first calculated by a heuristic for example.

The method consists, at each step, in decomposing a node that represents the solutions space associated with a partition of disjunctive subsets of lower sizes. The evaluation of the nodes allows to eliminate the branches that do not contain the optimal solution. So, before exploring a node, a lower bound of the objective function to the associated problem is calculated. If this value is higher than UB, the node and all the branches obtained from it are eliminated. If the node is a leaf, the exploration is stopped; otherwise the node is kept and UB is updated by the associated value of the objective function.

The quality of the method depends on the upper and lower bounds and the computation time. There are many strategies for selecting the nodes. The best known are the depth first where the most recently created node is selected, the breath exploration which consists to explore all the nodes of a level, the progressive method where priority is given to the most promising node (with the best evaluation).

Here are some references using the branch-and-bound procedure for production scheduling problems with limited resource availability: Souissi [Sou05], Lorigeon *et al.* [LBB02], Canon *et al.* [CBB03], Mauguière *et al.* [MBB03a,MBB03b,MBB05], Chen [Che06,Che07], Kacem and Chu [KC08a], Kcem *et al.* [KCS08] treat the single machine problem. Gharbi and Haouari [GH05], Mellouli *et al.* [MSCK09] investigate the parallel machines problem. Blazewicz *et al.* [BFKPS00], Kubiak *et al.* [KBFBS02] are interested in the flow shop problem. Allaoui [All04], Allaoui and Artiba [AA06], Kaabi [Kaa04] study the hybrid flow shop problem. Aggoune [Agg02], Mauguière *et al.* [MBB03a,b] tackle the job shop problem.

In our mathematical modeling and column generation approaches, we use the branch-and-bound procedure provided by ILOG CPLEX. For the first one, the objective is to find the optimal solutions to the problems integrating all the constraints and the variables. For the second one, the aim is to find the optimal solution associated to the problem defined by the improving columns added to the initial solution defined by the approximation methods that we develop.

2.1.2 Dynamic programming

Dynamic programming is a method for solving optimization problems, whose objective function has the property of decomposability (Gondran and Minoux [GM84]). These problems exhibit

the properties of overlapping subproblems and optimal substructure and their solution takes much less time than naive methods. It is proposed by Bellman [Bel57] in the context of a search for a shortest path in a graph.

The idea is to transform the resolution of one problem P to the resolution of sub-problems (P_0, P_1,..., P_n) related by a recurrence relationship on the value of the objective function. The information obtained during the subproblems resolution P_0,..,P_{k-1} are used to optimally solve the subproblem P_k. Thus, to obtain the optimal solution of the problem P, it is sufficient to go backward (P_n,.., P_0) through the set of taken and stored decisions of the resolution of each subproblem, which can cost high in time and memory.

As examples of using dynamic programming for scheduling with limited resource availability, Souissi [Sou05], Lee [Lee96], Sadfi [Sad02], Kacem *et al.* [KCS08] study the single machine problem, Lee [Lee96], Lee and Liman [LL93], Mellouli *et al.* [MSCK09] deal with the parallel machines problem, Lee [Lee97], Allaoui *et al.* [AAR03, AAER06], Kubzin et Strusevich [KS05, KS06] tackle the flow shop problem.

In our column generation approach, we integrate a dynamic programming procedure to search for the columns improving the model defined by the initial solution found by our approximation methods.

2.1.3 Linear programming

The linear programming is an important field of optimization for several reasons. Many practical problems in operations research can be expressed as linear programming problems.

The linear programming is a generic approach based on mathematical modeling of combinatorial optimization problems, where the constraints, expressed as inequalities and the objective function as an equation, are linear regarding the decision variables. So linear programming problems determine the way to achieve the best outcome (such as maximum profit or lowest cost) given some list of requirements represented as linear equation.

Geometrically, the linear constraints define a convex polyhedron which is called the feasible region. Since the objective function is also linear, hence a convex function, all local optima

are global optima. The linearity of the objective function also implies that the set of optimal solutions is the convex hull of a finite set of points - usually a single point.

The linear program is infeasible if the feasible region is empty (the constraints contradict each other). It is unbounded if the polyhedron is unbounded in the direction of the objective function.

When the decision variables are real, the variables of the linear programs are continuous, and are polynomial (algorithms of Khachiyan [Kha79] and Karmarkar [Kar84]). In practice, the most used algorithm is the simplex algorithm (Dantzig [Dan51]) although its theoretical complexity is exponential.

However, the majority of scheduling problems, which are NP-hard, can not be solved to optimality by programs with integer variables or mixed (real and integer). Their relaxation into continuous linear programs allows to obtain lower bounds for minimization problems.

2.1.3.1 Relaxation techniques

They have a double role in the resolution of combinatorial optimization problems. The principle of these methods is to relax a number of constraints (examples: variables integrity, resource constraints, precedence constraints between operations) in order to make the resolution of the problem easier. They provide good lower bounds to increase the efficiency of branch and bound methods. Fisher *et al.* [FLLRK83] proposed for instance an approach for the job shop based at first on the relaxation of the resource constraints; then on the relaxation of the jobs routings. For the small problems on which it is tested, it gives good results.

2.1.3.2 Column generation

In Barnhart *et al.* [BJNSV98], it is mentioned that the successful resolution of large-scale mixed integer programming problems (MIP) requires formulations whose linear programming (LP) relaxations give a good approximation of the convex hull of feasible solutions (as column generation is performed with the relaxation of the LP program). It is also mentioned that, in column generation, sets of columns are left out of the LP relaxation because there are too many columns to handle efficiently and most of them will have their associated variables equal to zero in an optimal solution anyway.

The principal of the column generation approach is the following: Starting from an initial set of columns that corresponds to a feasible solution to the problem, columns are iteratively added to the reduced problem (based on the LP program relaxation) thanks to a pricing problem that constructs feasible schedules corresponding to improving columns. The reduced problem is reoptimized at each iteration. The process is stopped when no column is added. If the solution to the reduced problem is integer, it is also the solution to the MIP; otherwise, the integrality property of the variables is introduced to the reduced problem that is then solved by a branch-and-bound procedure or a branch-and-price procedure.

In Lancia *et al.* [LRS07], a compact formulation of a model is an equivalent formulation in which the exponentially many constraints are replaced by a polynomial number of new constraints (after introducing an exponential number of new variables).

In Barnhart *et al.* [BJNSV98], considering formulations with a huge number of variables are listed may be due to the following reasons:

- When a compact formulation of a MIP may have a weak LP relaxation, the relaxation can be tightened by a reformulation that involves a huge number of variables,
- A formulation with a huge number of variables can eliminate the symmetry that can exist in the structure of a compact formulation of a MIP. This symmetry causes a poor performance of branch-and-bound,
- Column generation provides a decomposition of the problem into a master problem and a pricing problem. This decomposition may have a natural interpretation in the contextual setting allowing for the incorporation of additional important and complex constraints,
- A formulation with a huge number of variables may be the only choice.

2.1.3.3 Polyhedral approach - Cutting plan

It is widely used, for instance for the resolution of the traveling salesman problem and provides good results. The principle of the algorithm is to calculate, at the first step, the solution by the resolution of a linear program whose constraints are a subset of constraints defining the polyhedron. The second step consists in analyzing the solution to determine the constraints of the problem that are not satisfied. These constraints are added to the linear program and a new solution is calculated. The difficulty consists in finding unsatisfied constraints (for a solution).

Here are some references using linear programming for production scheduling with limited resource availability: Souissi [Sou05] for the single machine problem and Blazewicz *et al.* [BDFKS00, BDODM03] for the parallel machines problem.

In our modeling approach, integer linear and mixed integer linear programs are proposed to show how to deal with resource unavailability periods and their flexibility. Their results are used in the approximation and column generation approaches. Linear programming is also used in the column generation to elaborate the primal and dual programs.

2.2 Approximation methods

The approximation methods (heuristics) represent an interesting alternative to exact methods for solving NP-hard combinatorial optimization problems. Indeed, they can provide good solutions at low cost. They are of performance guarantee if it is possible to quantify the gap between the best provided solution and the optimal solution.

2.2.1 Construction heuristics

The construction heuristics are methods that iteratively build a solution. Most of them are greedy algorithms. The most commonly used methods are the list algorithms which principle is to sort the list of operations according to a decision strategy called *dispatching rule* such as SPT (Shortest Processing Time), EDD (Earliest Due Date), FIFO (First In First Out). In general, the generated schedules are either active or non-delay.

A dispatching rule is a rule that define priorities between all the jobs that are waiting for processing on a machine. The priority scheme may take into account the jobs and the machines attributes, and the current time. Whenever a machine becomes idle, a dispatching rule inspects the waiting jobs and selects the one with the highest priority. Research in dispatching rules has been active for several decades, and many rules have been developed and studied in literature.

Dispatching rules can be static or dynamic rules. Static rules are not time-dependent. They are a function of the job data, and/or the machine data; whereas dynamic rules are time-dependent.

Dispatching rules can be local or global. A local rule uses only information on the queue where the job is waiting or the machine (or workcenter) where the job is queued. A global rule may use information of other machines, such as the processing time of the job on the next machine on its route or the current queue length at that machine. As there are many basic dispatching rules, we present only few of them:

- SPT: Shortest processing time first rule (developed by Smith (1956)) sequences the jobs in non-decreasing order of their processing times. We list bellow some references using the SPT rule for production scheduling with resource unavailability periods, the considered problem is given in brackets: Lee [Lee96], Lee and Liman [LL92], Graves and Lee [GL99] (single machine), Kaspi and Montreuil [KM88], Liman [Lim91], Lee [Lee96] , Lee and Liman [LL93] (parallel machine), Adiri *et al.* [ABFRK89] (flow shop).

- WSPT: The weighted shortest processing time first rule is a generalization of SPT rule. Whenever a machine is available, the job with the highest ratio of weight w_i over processing time p_i ($\frac{w_i}{p_i}$) is scheduled next. This rule tends to minimize the weighted sum of the completion times, that is $\sum w_i C_i$ (Lee [Lee96] for the single machine problem).

- EDD rule: Earliest Due Date rule is due to Jackson [Jac55]. Whenever a machine becomes idle, the job with the earliest due date is selected to be processed next. This rule tends to minimize the maximum lateness among the jobs waiting for processing (Lee [Lee96], Graves and Lee [GL99], Li and Cao [LC95] for the single machine problem).

- LPT rule: Longest processing time rule orders the job in decreasing order of their processing times. When there are machines in parallel, this rule tends to balance the workload over the machines. The jobs with short processing times are kept for later to balance the workload. After the assignment of jobs to machines has been determined, the jobs on any given machine can be resequenced without affecting the workload balance. As examples of applications of LPT rule, Lee [Lee96] studies the single machine problem, Lee [Lee91], Lin *et al.* [LYH98] tackle the parallel machines problems, Allaoui [All04] deals with the hybrid flow shop problem.

- LRP rule: Longest Remaining Path (Liu and Sanlaville [LS95a,b, LS97] for the parallel machines problem).

In our approximation approach, we develop heuristics that construct a schedule based on various decision strategies. These strategies are defined relatively to how job operations and/or machines are prioritized, and how conflicts between jobs operations and machine unavailability periods are managed.

2.2.2 Decomposition heuristics

They consist in decomposing the problem into several subproblems. Among these methods are:

- *The hierarchical decomposition* (Erscher *et al.* [EM85]) which decomposes the problem into several levels, whose decisions become constraints for the lower levels,
- *The temporal decomposition* (Portmann [Por88]), which is used for dynamic scheduling problems. Subsets of available operations before the date T_1 are scheduled and some operations are included in the partial sequence. The remaining operations and the operations that become available between the dates T_1 and T_2 are grouped and then scheduled, and so forth,
- *The spacial decomposition* (Portmann [Por88]) which consists in decomposing the workshop in several workshops with a minimum of moves between them; scheduling operations in each workshop; and finally coordinating the whole.

2.2.3 Improving heuristics

The principle of these methods is not to build an initial schedule but to modify, starting from an initial solution, the result of a feasible schedule to improve the value of the objective function. Most of these methods use the notion of solution neighborhood. It therefore consists in exploring neighbor solutions of a given solution and select one of them to continue the exploration process. At each step, the chosen solution does not necessarily improve the value of the objective function but may allow to escape from local minima.

Improving heuristics can improve the sequences resulting from construction or decomposition heuristics. In particular, metaheuristics are widely used for solving combinatorial optimization problems. Their success is due to the fact that they can integrate different practical constraints

of the problems, they are easy to implement, and they provide good solutions. When sufficient knowledge about the search space is available a priori, one can often exploit that knowledge (inference) in order to introduce problem specific search strategies for finding solutions of higher quality.

The idea of a classical algorithm of iterative improvement is the following: it starts from an initial configuration. It then tries an elementary modification, called *movement*, and it compares the values of the objective function, before and after this modification. If the change leads to an improvement of this function, it is accepted, and the obtained configuration, which is the neighbor of the current solution, is the starting point for a new iteration. Otherwise, it comes back to the previous configuration before trying again. The process is repeated until every modification makes the result worse. In general, the algorithm does not lead to the global optimal solution; but only to a local optimum; which constitutes the best accessible solution depending on the initial assumptions.

To improve efficiency of the algorithm, it may be applied several times, modifying each time the initial assumption chosen randomly; and in the end, the best solution within the local minima is selected. But this may considerably increase the computation time without any guarantee of reaching the global optimum.

To escape from the local optimum, and explore more promising regions of the solutions space, more promising movements, which degrade the solution, may be allowed from time to time. To avoid the divergence of the process, a control mechanism of these degradations, depending on each heuristic, is added.

Among the techniques that have proved their efficiency in solving combinatorial optimization problems are the genetic algorithms (Holland [Hol75]), simulated annealing (Kirkpatrick *et al.* [KGV83]), tabu search (Glover [Glo86]), ants colony (Colorni *et al.* [CDM91]), local search, hill climbing. The interested reader will find more details on these different methods in Taillard *et al.* [TGGP01], Dréo *et al.* [DPST03] and Talbi [Tal09].

2.2.3.1 Simulated Annealing

The Simulated Annealing method, developed by Kirkpatrick *et al.* [KGV83], is inspired from the annealing process to solve optimization problems: the objective function of the problem, similar to material energy, is then minimized, introducing a fictitious temperature, which is in this case a control parameter of the algorithm.

In practice, the technique uses the algorithm of Metropolis, which allows to describe the behavior of a system in thermodynamic equilibrium under a temperature $Temp$: starting from a given configuration (feasible solution), an elementary modification is made to the system; if this transformation decreases the objective function (or energy), it is accepted; otherwise if it increases by Δf the objective function, it can be accepted with a probability equal to $\exp(-\frac{\Delta f}{Temp})$. The process is repeated, with a constant temperature, until the thermodynamic equilibrium is reached. After at least one modification, the temperature is reduced, before performing new iterations of transformations. This process is empirical.

The drawback of simulated annealing is the tuning of parameters (the reduction function of the temperature, the number of iterations for each temperature t, the initial temperature, where these three parameters represent the cooling scheme, the configuration and the neighborhood codings, the efficiency of the routines for generating a neighbor, and the computation time). The advantage is the flexibility regarding the problem evolutions and the easiness for implementing.

Blazewicz *et al.* [BBFKS01] uses a simulated annealing method to solve the flow shop problem with resource availability constraint.

Algorithm 1 describes the general scheme of the simulated annealing heuristic.

2.2.3.2 Tabu Search

The principal of the Tabu Search, proposed by Glover [Glo77], is to cleverly explore solutions space of the problem by avoiding getting stuck in a local optimum, thanks to two specific strategies. The first strategy is the *intensification* which forces the search in the most promising areas of the solutions space. The second strategy is the *diversification*, which guides the search in

Algorithm 1 The simulated annealing algorithm

Begin
Define an initial solution s
Evaluate this solution: $f(s)$
Initialize temperature $Temp$
while a stopping condition is not satisfied
Select s' a neighbor of s
if s' is better than s or $U \leq \exp(-\frac{\Delta f}{Temp})$ ($U \in [0,1]$: random uniform number)
then s' is the new value of s
end if
Decrease the temperature $Temp$
end while
end

new regions of the space.

Its principal particularity is the use of mechanisms inspired from the human memory. In opposition to the tabu search, the simulated annealing does not memorize the previously explored configurations; and then it is not able to learn from the past. However, the memory modeling induces multiple degrees of liberty which make difficult a rigorous mathematical analysis.

Indeed, it is the only metaheuristic used to solve optimization problems that works with a memory or a set of memories: the *explicit memory* that allows to save the solutions found during the search process and which is the basis of the diversification strategy, and the *attributive memory* which saves attributes such as operations permutations which allows to move from a solution to another.

Tabu search works with only one current *configuration* (solution) at a time. At first, an initial solution must be provided. While a stopping criterion is not met, this solution is progressively improved. This condition can be a fixed number of iterations or a fixed number of iterations without improvement of the solution.

The improvement process, applied at each iteration, consists in first associating to the current solution a neighborhood corresponding to a region of the solution space by applying an operation called *movement*. Hence the neighborhood is the set of accessible configurations in only one elementary movement from the current solution. It is important to well choose the

definition of the movement from those allowed; because only a part of the eventually huge *neighborhood* will be considered. It can be reduced by establishing a list of candidates or randomly extract un subset of neighbors of a fixed size.

The objective function is evaluated for each configuration of the neighborhood. The configuration selected is the one corresponding to the best value of the objective function. A configuration less good than the current one can be accepted if it meets a criterion. Thanks to this particularity, the method can avoid getting stuck in local mimima. The attributes of this movement are saved in the tabu list $Tlist$ for the $|Tlist|$ succeeding iterations (the movements are saved in the form $(new \rightarrow current)$; which are the opposite of the last movements $(current \rightarrow new)$.

This list avoids going back to solutions already visited in a recent past (cycling). However, the tabu status of a solution can be eliminated if some conditions expressed by the *aspiration criterion* are satisfied. One of the most used aspiration criteria is the global aspiration criteria which consists in selecting a tabu movement if it allows the improvement of the best value of the objective function found so far.

The tabu list can be explicit or attributive depending on the used restrictions. The choice of the list type depends on the considered problem.

For some optimization problems, the tabu search gives good results. Moreover, in its basic form, the method contains less tuning parameters than the simulated annealing, which make it simple for use. However, some mechanisms such as the intensification and the diversification, bring a notable complexity.

Algorithm 2 describes the general scheme of the tabu search.

Aggoune [Agg02] develops a tabu search approach to solve the flow shop and the job shop problems with resource availability constraints.

Algorithm 2 The tabu search algorithm

Begin
    Define an initial solution s
    Evaluate this solution: calculate $f(s)$
    while a stopping condition is not satisfied
        Find the best allowed movement in the neighborhood of s
        Update the memory structures
        Construct the solution s' associated with the selected movement
    end while
end

2.2.3.3 Genetic Algorithms

Unlike local search algorithms such as Simulated Annealing and Tabu Search which need only one feasible solution, the Genetic Algorithms consider a population of feasible solutions called *individuals*.

Genetic Algorithms, by Holland [Hol75], are evolutionary algorithms which build solutions by combining others. A set of a defined number of points in the search space, chosen randomly or by a heuristic, constitutes the initial population; each *individual* (*chromosome*) of the population has a performance, which measures his degree of adaptation to the wanted objective. The algorithm consists in evaluating progressively, by successive generations, the composition of the population, by maintaining its size constant and finding strong individuals. Among the generations, the objective is to globally improve the individuals performance. They simulate the natural process of species evolution by adopting two laws that characterize them: the transmission of hereditary characters at reproduction, and the law of survival within the population according to Darwin's theory:

- The selection, which promotes the reproduction and the survival of the most performant individuals,
- The reproduction, which allows the brewing, the recombination and the variations of hereditary characters of the parents, to form offsprings of a new potential.

So the *chromosome coding* must be defined at first. Some of them are of type: binary vectors (Goldberg [Gol89]), permutations (Reeves [Ree95]), direct (explicit representation of the solution) and indirect (contains only characteristics to build the solution to which it corresponds).

Once the initial population is obtained, a strength called *fitness* is assigned to each individual to quantify its importance within the population. Then, the initial population is improved, until a stopping condition is met (generally a fixed number of iterations). The transition from a generation to the next is performed in four phases: a selection phase, a reproduction phase (or variation), a phase of performances evaluation and a replacement phase.

The purpose of the *selection phase* is to drive the population towards increasingly better solutions by discarding out bad solutions. It designs the individuals which participates to the reproduction. They are chosen, eventually several times, a priori as often as they have a good performance. Some of the selection modes are:

- *The tournament selection:* it consists in selecting the next generation by conducting a number of tournaments between individuals.
- *The proportional (Roulette-wheel) selection (Goldberg [Gol89]):* it consists in assigning a selecting slot to each individual according to its fitness.
- *The ranking selection:* the individuals are sorted (ranked) by fitness and the selection is performed as in the proportional selection.
- *The steady-state selection:* It consists in creating and inserting, in the population, a small number of offsprings at each iteration; then the worst individuals are discarded.

The reproduction phase consists in applying variation operations on the selected individuals to create new ones; the most used operators are *crossover* and *mutation*. The structure of variation operators tightly depends on the representation chosen for individuals.

The crossover operator, which consists in combining the genes of two chromosomes and create new individuals, is applied at first. The most simple one is the 1-point crossover denoted 1X (Goldberg [Gol89]): the crossover point is randomly chosen; it decomposes the two parents C_1, C_2 into two sub-chains C_{11}, C_{12} and C_{21}, C_{22}. The first offspring is obtained by the concatenation of C_{11} and C_{22}, the second one by C_{21} and C_{12}. There are other crossover operators in literature. For example, for permutations crossovers, there are OX for Order Crossover (Oliver *et al.* [1987]), LOX for Linear Order Crossover (Falkenauer and Bouffouix [1991]), PMX for Partially Mapped Crossover (Goldberg et Lingle [1985]) and ERX for Edge Recombination

Crossover (Whitley [1989]).

The second operator involved in the generation process of the new individuals is the mutation operator. It is a perturbation operator of randomly chosen chromosomes that modifies some of their characteristics. In the case of a permutation coding, the mutation may be positions exchanging of some genes. This prevents a too fast convergence to a sub-population limited to these genes and a loss in the quality of the solution.

The choice of coding, selection, operators depend on the problem. The performance of the Genetic Algorithm depends on its characteristics: the population size, the crossover and mutation probabilities.

Finally, the replacement phase is performed. It consists in selecting the new population. The algorithm is interrupted after a fixed number of generations, according to a stopping criterion.

Algorithm 3 describes the genetic algorithm.

Algorithm 3 The genetic algorithm

Begin
  Define a coding and generate an initial population
  Evaluate each individual of the population
  while a stopping condition is not satisfied **do**
    Select individuals for recombining
    Apply variation operators (crossover, mutation) on the selected individuals
    Evaluate the performance of the new individuals
    Replace individuals to get the new population
  end while
end

As examples of applications of genetic algorithms to production scheduling under machine availability constraints, Aggoune [Agg02] and Kaabi [Kaa04] study the flow shop problem, Aggoune [Agg02], Harrath [Har03] tackles the job shop problem and Levitin [Lev00], Zribi [Zri05], Gao *et al.* [GSS06], Chan *et al.* [CWC06] deal with the flexible job shop problem.

2.3 Conclusion

In this chapter, we present the techniques that are used to solve scheduling problems. For the two classes of methods: exact and approximation, the most representative techniques are described. We focus on those that are used in our study. For most of the methods described, some references are given for production scheduling problems with resource unavailability periods.

Chapter 3

State of the art of production scheduling problems with resource unavailability periods

This chapter is dedicated to results that are present in the literature for production scheduling problems with resource availability constraints. Section 3.1 is devoted to the production problems taking into account fixed unavailability periods. Results on flexible unavailability periods are presented in Section 3.2. The flexibility is on the starting dates of unavailability periods; they may vary in an intervals defined by earliest and latest starting dates.

3.1 Problems with fixed availability constraints

3.1.1 Single machine problems

Non-preemptive case

Souissi [Sou05] studies the $1||\sum w_i C_i$ problem. Lower bounds and mathematical properties of the problem are given. An integer linear program, a branch-and-bound method and a dynamic programming method are proposed. The tests show that the dynamic programming method performs better than the branch-and-bound, which is better than the integer linear program. In addition, the author considers the problem of makespan minimization with availability dates of operations.

Sadfi *et al.* [SPRBF05] tackle the $1||C_{max}$ with a maintenance period. An approximation algorithm, with a worst case error bound of $\frac{3}{17}$, is proposed to solve the problem. It is shown that the bound is tight through an example.

Kacem and Chu [KC08a] study the $1||\sum w_iC_i$ with one availability constraint. A branch-and-bound algorithm, based on a set of improved lower bounds and heuristics, is proposed. It is able to solve instances of 6000 jobs in a reasonable computation time.

Kacem *et al.* [KCS08] investigate the $1||\sum w_iC_i$ with one unavailability period. The authors design three exact methods for solving such problem: a branch-and-bound method based on new properties and lower bounds, a mixed integer programming model, and a dynamic programming method. These approaches solve problems of 3000 jobs within a reasonable computation time. The numerical experiments show the complementarity of the dynamic programming method and the branch-and-bound method.

Resumable case

Lee [Lee96] studies the single machine problem for different performance measures and identifies some polynomial problems. In particular, the author shows that for the $1|rs|C_{max}$ problem, any sequence is optimal. The author also shows that scheduling operations under SPT rule allows to optimally solve the $1|rs|\sum_i C_i$ problem. Similarly, the $1|rs|L_{max}$ and $1|rs|\sum_i U_i$ are optimally solvable by respectively the EDD rule and a modification of the Moore-Hodgson rule.

The classical $1||\sum_i w_iC_i$ problem (i.e., without machines unavailability) can be optimally solved by sequencing jobs with SWPT rule. But when introducing availability constraints, Lee [Lee96] proves that the problem becomes NP-hard in a weak sense even if $w_i = p_i$ for all $i = 1, \cdot\cdot, n$. In addition, a dynamic programming algorithm, and several heuristics are proposed to solve the problem in the case of one unavailability period.

Lorigeon *et al.* [LBB02a] study the single machine problem, with one unavailability period, considering heads and tails on operations. The authors propose a lower bound, two upper bounds, a branch-and-bound method. Note that when the machine is continuously available, the problem is strongly NP-hard.

Wu and Lee [WL03] proposes an algorithm for scheduling linear deteriorating jobs on a single machine (time-dependent scheduling) with one availability constraint and the makespan minimization.

The $1, \alpha_1|r_i, \beta_1(M_1^k)|max\{C_i + q_i\}$ problem ((single machine problem with k unavailability period on the machine that are crossable or non-crossable; jobs are resumable or non-resumable and have release dates and latencies)) with $\alpha_1 \in cr/ncr$ $\beta_1 \in rs, nr, rs/nr$ is strongly NP-hard. Indeed the same problem without unavailability periods is strongly NP-hard.

For the $1|rs|max_{1\leq i\leq n}\{C_i + q_i\}$ problem, a simple modification of Carlier's branch-and-bound algorithm (1982) allows Canon *et al.* [CBB03] to solve the problem. So problems with up to 500 jobs are solved in less than 1 minute in the worst case and in less than 1 second on the average.

To efficiently solve the problem $1, cr/ncr|rs|max\{C_i + q_i\}$, Mauguière *et al.* [MBB05] integrates in the resolution method of the $1|nr|max\{C_i + q_i\}$ problem a branch-and-bound method that solves the $1|rs|max\{C_i + q_i\}$ problem.

For minimizing the arrival time of the last delivery batch to the distribution center ($\sim C_{max}$) in batch production on a single machine, Wang and Cheng [WC06] provide a polynomial algorithm.

Kacem and Chu [KC08b] study the $1|rs|\sum w_iC_i$ with one unavailability period on the machine. New properties of the worst-case performance of the WSPT heuristic and a tighter approximation of the worst-case error are given. The worst-case bound is equal to 2 under some conditions. The results complete those of Lee [Lee96].

Non-resumable case

Adiri *et al.* [ABFRK89] study the $1|nrs|\sum_i C_i$ problem. The deterministic and stochastic contexts on unavailability are both considered. In particular, in the deterministic case, it is proved that the problem is NP-hard in the weak sense, even if there is only one unavailability period on the machine.

Lee and Liman [LL92] develop a simpler proof for NP-hardness of the deterministic problem considered by Adiri *et al.* [ABFRK89]. The authors also show that the rule has a performance guarantee equal to $\frac{9}{7}$.

Lee [Lee96] shows that the $1|nrs|C_{max}$ problem is NP-hard in a weak sense, as soon as one unavailability period is considered. The author also proves that an algorithm sequencing operations according to LPT rule has a relative error equal to $\frac{1}{3}$.

In the same paper, it is shown that the $1|nrs|L_{max}$, $1|nrs|\sum_i U_i$ and $1|nrs|\sum_i w_i C_i$ problems are NP-hard in a weak sense. In addition, the EDD algorithm solves the first problem with a relative error equal to p_{max} (longest processing time) and the second problem can be solved by Moore-Hodgson rule with a relative error equal to 1. For the third problem, Lee [Lee96] shows that the performance ratio of the SWPT algorithm may be arbitrarily high, even if $w_i = p_i$ for any $i = 1, \cdot\cdot, n$.

Sadfi [Sad02] studies the same problem as Lee and Liman [LL92]. For the resolution, the author develops the MSPT heuristic (Modified SPT) with a performance guarantee of $\frac{19}{17}$ improving the best bound $\frac{9}{7}$ found so far in the literature. The MSPT rule consists in improving result given by SPT by swapping a job ordered before the unavailability period with another job sequenced after the unavailability period. The author also develops a dynamic programming algorithm of pseudo-polynomial complexity leading to the optimal solution of the problem.

Leon and Wu [LW92] propose a branch-and-bound method for the single machine problem with machine unavailability. The algorithm is an adaptation of the branch-and-bound of McMaron and Florian [MF75] and solves problems of 50 operations. Balas *et al.* [BLSV98] consider the single machine problem with precedence constraints and due dates (deadlines), which generalizes the problem studied by Leon and Wu [LW92], and easily solve their benchmarks in terms of computation time.

Wang and Cheng [WC06] propose a heuristic for batch production on a single machine. This heuristic has a worst-case error bound of $\frac{1}{2}$. The authors show that this bound is tight. Production and job delivery are considered at the same time. The objective is the minimization of the arrival time of the last delivery batch to the distribution center (what is equivalent to

C_{max}). Moreover, one vehicle with at most K-job capacity is available to deliver the jobs in a fixed transportation time to a distribution center.

For the $1|nrs|\sum C_i$, Chen [Che06] assumes that the machine availability is limited due to periodic maintenance activities. Several maintenance periods define a periodic maintenance schedule; and each maintenance period is scheduled after a periodic time interval. A branch-and-bound algorithm is proposed to optimally solve the problem; and a heuristic is designed to solve large sized problems.

Gawiejnowicz [Gaw07] tackles the single machine problem with n deteriorating jobs and k unavailability periods ($1 \leq k < n$). The author develops an algorithm for C_{max} minimization. The starting date $t_i = \alpha_i t$, where $\alpha_i > 0$ is the deterioration rate and $t > 0$ is the time (t is applied instead of the starting process time of J_i). The author proves that the problem is NP-hard in the ordinary sense if there is only one unavailability period; otherwise it is strongly NP-hard.

Chen [Che07] considers a periodic maintenance scheduling problem on a single machine in a textile company. The processing times and due dates are integer values. The author develops a near optimal heuristic and an optimal branch-and-bound algorithm to minimize T_{max}. The results show that the heuristic is accurate and efficient.

Resumable / Non-resumable case

To solve the $1, cr|rs/nrs|max\{C_i + q_i\}$ problem, Mauguière *et al.* [MBB03a] propose a branch-and-bound algorithm. Most of instances with up to 100 operations are solved, though some smaller instances seem to be intractable for the method.

To solve the $1, cr/ncr|r_i, rs/nrs(M_1^k), d_i|max\{C_i + q_i\}$ problem, Mauguière *et al.* [MBB03b] develop a branch-and-bound procedure. Another solution method is proposed by Mauguière *et al.* [MBB05]. The strongly NP-hard problem $1|pmtn, r_i, d_i, q_i|max(C_i + l_i)$ is solved by the authors with an approximation algorithm which is a modification of Schrages algorithm (1971); the latency duration $l_i = max\{q_i, Kd_i\}$, where K is a constant number. As the problems with rs, $cr/ncr|rs$, nrs, $cr|rs/nrs$ and $pmtn$ are particluar cases of the problem with $cr/ncr|rs/nrs$,

the $1, cr/ncr|r_i, rs/nrs, d_i|max\{C_i + q_i\}$ problem is NP-hard too, but Mauguière *et al.*[MBB05] propose an algorithm that solves it in reasonable time and accuracy.

3.1.2 Parallel machines problems

Non-preemptive case

Gharbi and Haouari [GH05] investigate the multiprocessor scheduling problem with non simultaneous machine availability times, release dates, and delivery times ($P, NC_{inc}|r_i, q_i|C_{max}$), where NC_{inc} indicates that the number of available machines is nondecreasing with time). The authors propose new lower and upper bounds, and a branching strategy based on a schedule coding as a permutation of jobs. Introducing a semi-preemptive lower bound, based on max-flow computations, in a branch-and-bound algorithm, yields very promising performance. A semi-preemptive schedule, which concept is introduced by Haouari and Gharbi [HG03], is defined as a schedule such as the fixed parts of the jobs are constrained to start and to finish at fixed times with no preemption, whereas the free parts can be preempted. This method can solve instances of 700 jobs and 20 machines within a reasonable CPU time. It can also be used to solve large instances of two important particular cases $P, NC_{inc}||C_{max}$ and $P|r_i, q_i|C_{max}$.

Mellouli *et al.* [MSCK09] address the $P||\sum C_i$ with a planned maintenance period on each machine. Three exact methods (mixed integer linear programming methods, a dynamic programming based method and a branch-and-bound method) and several constructive heuristics are proposed. Moreover, the authors give dominance properties, a lower bound, and two branching schemes for the branch-and-bound method.

Resumable case

Schmidt [Sch84] studies the $P_m|prmp, rs|C_{max}$ problem. The author proposes an algorithm of $O(n + m \log n)$ complexity to build feasible preemptive schedules in the case where all machines are available during an arbitrary number of periods. In Schmidt [Sch88], release and due dates are taken into account and it is proved that the problem is tractable in $O(n \log nm)$ steps. When no release dates are imposed, minimizing the largest delay can be obtained in a time proportional to $O(nm \log n)$.

The $1|rs|\sum_i w_i C_i$ problem is NP-hard implies that the $P_2|rs|\sum_i w_i C_i$ is also NP-hard.

Kaspi and Montreuil [KM88] and Liman [Lim91] prove that scheduling jobs with the SPT rule is an optimal order for the $P_m|rs|\sum_i w_i C_i$ problem in the context of parallel machines that are not available at time zero.

Lee [Lee96] states that the $P_m|rs|C_(max)$ problem which is an extension of $P_m||C_{max}$ problem is also NP-hard. The author also develops a dynamic programming algorithm to optimally solve the problem considered by Kaspi and Montreuil [KM88].

Lee [Lee91] considers the identical parallel machines problem assuming that these machines are not available at time zero, with the makespan minimization. The author proposes a heuristic with a relative error of $\frac{1}{2}$ based on the LPT rule, then an improvement of this algorithm with error equal to $\frac{1}{3}$.

Lin *et al.* [LYH98] are interested in maximizing the smallest jobs completion date in an environment of m parallel machines unavailable at time zero. The authors show that the LPT order has a worst-case error equal to $\frac{2m-1}{3m-2}$. The considered criterion helps to balance the workload on the machines.

Lawler and Martel [LM89] solve the problem $Q_2|pmtn, rs|\sum w_i U_i$. Pseudo-polynomial algorithms of complexity of $O(\sum_i w_i n^2)$ or $O(n^2.t_{max}))$ are proposed using dynamic programming.

Sanlaville [San95] investigates the problem of scheduling preemptive independent jobs, on identical processors, so as to minimize the maximum lateness L_{max}. The author suggests a nearly on-line priority algorithm with an absolute error less than or equal to $(m - \frac{1}{m})t_{max}$ (where t_{max} is the maximum of the starting dates of the jobs) if the machines availability follows a constant pattern, and it is less than or equal to t_{max} if the machine availability represents an increasing zigzag pattern. The priority is calculated according to the Smallest Laxity First (SLF) rule, (the laxity or slack time is the difference between the due date of the job and its remaining processing time). The complexity of the method is $O(n^2.t_{max})$ and, in the case of a zigzag pattern and no release dates, the obtained solution is optimal. For the problem with release dates and due dates, a method of $O(n^3.t^3_{max}(\log n + \log t_{max}))$ complexity is implemented, if the number of changes of machine availabilities during any time interval is linear in length of

the interval. This algorithm is off-line.

Liu and Sanlaville [LS95a, LS97] study the parallel machine problem with resumable availability constraints considering the precedence constraints. For the $P_m, NC|prmp, chains|C_{max}$ problem (the problem with chains and arbitrary pattern of unavailability), the Longest Remaining Path (LRP) rule is used to solve the problem in a polynomial time by. The $P_{m2}, NC|prmp, prec|C_{max}$ (the problem of two parallel machines and arbitrary patterns of availability) can also be solved by LRP in case of arbitrary task precedence relations in time complexity $O(n^2)$.

Liu and Sanlaville [LS95a] show that using SLF rule on modified due dates, for the makespan minimization for inforest precedence graphs and increasing zigzag patterns, the results can be extended to minimization of L_{max}. Note that the modified due date is given by $d'_i = min\{d_i, d_{s(i)} + t_{s(i)}\}$, where index $s(i)$ is related to the successor job of J_i when it exists. SLF rule is also used for L_{max} minimization on the two machines problem with availability constraints, but with a different modification scheme.

Sheen and Liao [SL07] tackle the problem of scheduling n preemptive jobs on m machines with identical speed under machine availability and "eligibility" constraints for minimizing L_{max}. The problem is formulated into series of maximum flow problems by network flow technique. A polynomial time two-phase binary search algorithm is proposed to check the feasibility of the problem and if it is feasible to optimally solve it. Finally, it is proved that the time complexity of the algorithm is $O((n + (2n + 2K))^3 \log(UB - LB))$, where K is the total number of availability periods on all machines, and UB and LB are respectively upper and lower bounds provided by the algorithm for optimal L_{max}.

Blazewicz *et al.* ([BDFKS00], [BDODM03]) show that the parallel processors problem, with preemptive tasks, multiprocessor tasks and limited processors availability, becomes NP-hard in the strong sense in case of trees and identical processors. When the tasks form chains and are processed by identical processors with a staircase pattern (NC_{sc}) of availability, the problem can be solved in a low-order polynomial time for C_{max} and a linear programming approach is required for L_{max}. For the problem with independent tasks scheduled on uniform and unrelated

processors with arbitrary patterns of availability, the network flow and linear programming approaches are respectively proposed for the schedule length and maximum lateness criteria.

For the batch production problem on two parallel machines, in case only one processor has an unavailability period, Wang and Cheng [WC06] propose a heuristic to minimize the arrival time of the last delivery batch to the distribution center. This heuristic has a worst-case error bound of $\frac{2}{3}$.

Non-resumable case

Ullman [Ull75] is the first to address the minimization makespan on m parallel machines with availability constraints. In the case of non-scored, the author demonstrates that the problem is NP-hard in the weak sense.

Lee and Liman [LL93] study the $P_2|nrs|\sum_i w_iC_i$ problem assuming that one of these machines are no longer available from a date. The authors show that the problem of minimizing the sum of job completion dates is NP-hard in the weak sense and propose a dynamic programming algorithm, and an SPT based heuristic for its resolution. The performance guarantee of the heuristic is equal to $\frac{3}{2}$.

Mosheiov [Mos94] considers the same problem with the assumption that each machine is available only for a period. The author shows that for the $P_m|nrs|\sum_i w_iC_i$ problem, scheduling jobs under SPT rule is asymptotically optimal, when the number of jobs tends to infinity.

A problem similar to that of Mosheiov [Mos94] is the scheduling problem with time windows. In this problem constraints on the jobs availability are considered rather than machines. For more details, refer to Lei and Wong [LW91], and Kraemer and Lee [KL93].

Lee [Lee96] shows that the $P_m|nrs|C_{max}$ problem is NP-hard. The performance of two heuristics are also analyzed: SPT rule and scheduling list SL, which consists in assigning an operation (given any jobs processing order) to the machine that leads to the smallest completion date. The SPT and SL algorithms have respectively relative errors of $\frac{m+1}{2}$ and m. In the same paper, the author shows that the $P_2|nrs|\sum_i w_iC_i$ problem is NP-hard. A dynamic

programming algorithm is developed to solve efficiently the problem in case $w_i = 1$ for all i and the first machine is continuously available.

3.1.3 Flow shop

Non-preemptive case

Aggoune [Agg02] proposes two approaches for solving the problem. The first one is based on a list algorithm, the second one uses the geometric approach (two-job) (see also Aggoune and Portmann [AP06], the $F, NC_{win}|n = 2|C_{max}$ is polynomial and its complexity is at most equal to $O(k.m^4)$). These approaches are coupled with metaheuristics to improve their performances. The test results show that the two-job algorithm provides better results that the greedy one.

Cheng and Liu [CL03a] tackle the two-machine no-wait flow shop problem in which each machine can have an unavailability period. Algorithms are proposed for the cases where an unavailability period is imposed on only a machine and when unavailability periods on the two machines overlap. These algorithms improve the existing results and have a performance bound in the worst-case of $\frac{3}{2}$. In an another paper [CL03b], the authors develop an approximation scheme in polynomial time for these problems; that seem interesting only in theory since their complexity depends on a coefficient whose value depends on the wanted accuracy. The approximation algorithms presented in the paper [CL03a] are more efficient.

Resumable case

Lee [Lee97] studies the $F_2|rs(M_r)|C_{max}$ problem. The author shows that the problem is NP-hard in the weak sense whatever the machine concerned by the unavailability; and proposes pseudo-polynomial algorithms based on dynamic programming. The author also develops a heuristic with a performance guarantee of $\frac{3}{2}$ (resp. $\frac{4}{3}$); where the unavailability period occurs on the first machine (resp. the second), the relative error obtained by applying the Johnson's algorithm is equal to 1 (resp. $\frac{3}{2}$). In the same paper, it is demonstrated that for one or both machines unavailable at instant zero, the problem is optimally solved by Johnson's algorithm.

Lee [Lee99] considers the $F_2|rs, nr, sr|C_{max}$ problem. In the resumable case, it is shown that in case of an unavailability period on each machine, Johnson's algorithm is optimal when the unavailability periods are planned at the same date. In addition, when the machines are

unavailable at different dates, and even if the length of unavailability periods are equal, the problem becomes NP-hard in the weak sense.

Cheng and Wang [CW99] consider the two-machines flow shop problem with an unavailability period on each machine. The authors assume that the unavailability periods are consecutive and the operations are semi-resumable. A heuristic, of worst-case bound equal to $\frac{2}{3}$ is developed for makespan minimization in the non-resumable case.

Cheng and Wang [CW00] address the two-machine flow shop problem for the makespan minimization when the operations are resumable and there is an unavailability period on the first machine. The authors show that the worst-case bound equal to $\frac{1}{2}$ and found by Lee [Lee97] is tight; then they develop a heuristic with a performance guarantee equal to $\frac{4}{3}$.

Blazewicz *et al.* [BFKPS00] propose a parallel implementation of a branch-and-bound method for the makespan minimization in a two-machine flow shop problem where several unavailability periods by machine are considered and the operations are resumable.

Wang and Cheng [WC01] develop two heuristics, with a worst-case bound equal to $\frac{5}{3}$. The authors improve the results of Espinouse et al [EFP99] for the no-wait two-machines flow shop problem with availability constraints.

In Blazewicz *et al.* [BBFKS01], two constructive heuristics and a simulated annealing method are developed to solve the problem addressed in Blazewicz *et al.* [BFKPS00]. The first heuristic schedules the jobs between two consecutive unavailability periods following Johnson's rule, while the second one is based on a local optimization.

Braun *et al.* [BSS01a] study the stability of the schedules for a two-machine flow shop problem in the presence of an unavailability period on each machine. In particular, the authors show that for the makespan minimization with resumable operations, the Johnson's order is still dominant if the unavailability periods are sufficiently small.

Kubiak *et al.* [KBFBS02] also consider several unavailability periods in the two-machine flow-shop with resumable operations. The authors show that the makespan minimization is

strongly NP-hard, even if all unavailability periods concern only one machine. In case where unavailability periods are on the first machine, a heuristic with a performance guarantee equal to 2 and of complexity $O(n \times logn)$ is proposed; and it is shown that there is no heuristics with a performance guarantee for more than two unavailability periods, if at least one of them is on the second machine. In addition, after proving that scheduling jobs according to the Johnson's rule between two consecutive unavailability periods is optimal, the authors develop a branch-and-bound method to solve the problem of sequencing jobs on the machines in the different intervals of availability.

Breit [Bre04] studies the $F_2|rs|C_{max}$ with an unavailability period on the second machine. A relative error in the worst-case of $\frac{5}{4}$ is proposed; thereby improving the result given by Lee.

Breit [Bre06] tackles the $F_2|rs|C_{max}$ with n preemptive jobs and an unavailability period on the first machine. The author develops a polynomial-time approximation scheme to solve this problem; then it is extended to solve the problem where the unavailability period is on the second machine.

Wang and Cheng [WC07a] propose two heuristics of worst-case error bounds not longer than $\frac{2}{3}$ for the $F_2|rs(M_r), S_{ri}|C_{max}$ problem.

Wang and Cheng [WC07b] tackle the following permutation flow shop problem:
$PF2|rs(M_1), S_{ri}|C_{max}$ and present a polynomial-time approximation scheme for it.

Kubzin *et al.* [KPS09] study the $F_2|rs, srs|C_{max}$ problem with machine availability constraints. The authors propose a fast 32-approximation algorithm for the problem with several non-availability periods on the first machine and resumable operations. When there is one unavailability period and the operations are semi-resumable, a polynomial-time approximation scheme is presented.

Non-resumable case

Based on the work of Adiri *et al.* [ABFRK89], the problem $F|nrs|\sum C_i$(and so,
$F|nrs(M_r^k)|\sum w_iC_i)$ is NP-complete, because the $1|nrs|\sum C_i$ problem with only one unavailability period is NP-hard. The SPT rule leads to a tight relative error not greater than $\frac{2}{7}$ for

this problem. For fixed m, the SPT rule is asymptotic optimal if there is no more than one interval of non-availability for each machine (refer to Sanlaville and Schmidt [SS98]).

Cheng and Wang [CW99] study the $F_2|nrs(M_r^2)|C_{max}$ problem with two consecutive availability constraints. The authors develop a heuristic and show that it has a worst-case error bound of $\frac{2}{3}$.

For the $F_2|nrsM_r|C_{max}$ problem, Lee [Lee99] proposes a heuristic with a relative error equal to 1 when the unavailability period is imposed on the first machine. When the unavailability period is imposed on the second machine, the Johnson's algorithm has a relative error of 1.

Espinouse *et al.* [EFP01] study the two-machine no-wait flow shop problem with machine availability constraints and makespan minimization. The unavailability periods are known in advance. The authors prove that even if only one unavailability period occurs on one of the machines, the problem is NP-hard. It becomes NP-hard in the strong sense for arbitrary numbers of unavailability periods. They also propose heuristics to solve the problem.

Braun *et al.* [BLSS02] tackle the problem of minimizing the makespan in the two-machine n-job flow shop scheduling with k_1 non-availability periods on each of the two machines. This problem is binary NP-hard even if there is only one non-availability period either on the first machine or on the second one.

Ng and Kovalyov [NK03] address a deterministic two-machine flow shop scheduling problem with unavailability of one of the machines in a specified time period. The authors show that the two cases of the problem when the unavailability period is for the first or the second machine are equivalent to similar partition type problems. A generic fully polynomial time approximation scheme is developed. It has a time complexity of $O(\frac{1}{4}n^5)$.

Kubzin and Strusevich [KS05] investigate the two-machine flow shop scheduling problem with no-wait in process, and one of the machines has a maintenance activity of a length defined by a non-decreasing function that depends on the starting time of that maintenance. The objective criterion is the makespan minimization. A polynomial-time approximation scheme is

proposed to solve the problem.

Kubzin and Strusevich [KS06] consider the $F_2||C_{max}$ in which each machine has a maintenance activity, and whose duration depends on its starting time. The authors prove that the problem is binary NP-hard and is pseudo polynomially solvable by dynamic programming.

Semi-resumable case

Lee [Lee99] generalizes the complexity results given in [Lee97] to the case where the operations are semi-resumable. More specifically, the author shows that the makespan minimization is NP-hard in the weak sense whether the unavailability period is on a machine or the other $(F_2|srsM_r|C_{max})$. When the unavailability period is imposed on the first machine, a dynamic programming algorithm is developed, and the proof that the Johnson's algorithm has a relative error equal to 1 is given. When the unavailability period occurs on the second machine, Johnson's algorithm has a relative error equal to $max\{\frac{1}{2}, \alpha\}$, where α is the portion of the semi-resumable operation, interrupted by the unavailability period, to be reprocessed.

In the same paper, the author shows that when both machines have an unavailability period (which do not start at time instant zero), the problem is NP-hard in the weak sense, even if the starting and completion dates of unavailability periods are the same on both machines. In this case, the Johnson's algorithm has a relative error equal to α. Finally, if unavailability periods of both machines begin at instant zero, the Johnson's algorithm allows optimally solving the makespan minimization.

Resumable / non-resumable cases

Espinouse *et al.* [EFP99] are interested in the makespan minimization in the no-wait two-machine flow shop problem, with availability constraints. The authors show that the problem is NP-hard in the weak sense when only one unavailability period is considered. They develop a heuristic of performance guarantee equal to 2 in resumable and non-resumable cases. They demonstrate that when several unavailability periods are taken into account, the problem became strongly NP-hard; and there is no heuristic of a performance guarantee (whether operations are resumable or not).

Allaoui *et al.* [AAR03] tackle the makespan minimization for the two-machine flow shop problem whose first machine has unavailability period. The authors consider two scenarios: resumable and non-resumable operations. For both scenarios, they propose a dynamic programming algorithm. Moreover, they focus on studying the performance of the Johnson's algorithm. They establish the optimality condition and show that in the other cases its performance is bounded by 2.

Allaoui *et al.* [AAER06] study the two-machine flow shop problem for the makespan minimization with an unavailability period on the first machine and in the resumable and non-resumable cases. The authors propose an improvement of a dynamic programming model comparing to the one proposed by Lee [Lee97]. It reduces the computation of the optimal solution and proves that the bound is tighter.

3.1.4 Hybrid flow shop

Non-resumable case

Allaoui [All04] studies the two-stages hybrid flow shop problem with one machine at the first stage, and m machines at the second one. The author proposes a branch-and-bound method to solve small problems and three heuristics based respectively on a list algorithm, LPT rule and Johnson's rule. A study of the worst-case is made for the three heuristics. A simulation model is developed for the general hybrid flow shop problem.

Allaoui and Artiba [AA06] study the same problem. The authors assume that jobs have to wait between stages and preemption is not allowed (although the terminology of non-resumable case is used). They assume that each machine has at most one unavailability period. The optimization criterion is makespan. The problem is strongly NP-hard. They propose a branch-and-bound algorithm for small size problems. For problems of large sizes, they calculate the error bounds of a list algorithm, LPT algorithm and heuristic H proposed by Lee and Vairaktarakis (1994) for the hybrid flow shop without availability constraints. They prove that the performance of heuristic H, in the worst-case, is better than LPT one if and only if the number of machines in the second floor is greater than or equal to 4.

Jungwattanakit *et al.* [JRCW08] investigate the hybrid flow shop problem with n jobs, unrelated parallel machines at each stage, due dates, release dates and sequence and machine

dependent setup times. The authors formulate the problem in a 0-1 mixed integer program for combinations of C_{max} and $\sum U_i$. The preemption on job operations is not allowed and only one unavailability period is possible for each machine at time zero. Heuristics based on existing dispatching rules and well-known constructive heuristics for the flow shop problem with makespan minimization are proposed. Improvement methods that are polynomial and based on job shifting are used for the solutions; genetic algorithms are also proposed.

3.1.5 Job shop

Non-preemptive case

Aggoune [Agg02] studies the two-job shop problem for makespan minimization in the case of strictly non-preemptive operations. The author develops an extension of the geometric approach of Akers and Friedman [AF55] that can transform the initial problem in a search for a shortest path. The method is polynomial. It is based on a new characterization of vertices, on the introduction of additional vertices, and on taking into account the time during the scheduling. The algorithm is then extended to take into account any regular criterion, additional precedence constraints and release dates on operations.

Aggoune [Agg02] adapts the approximation methods he develops for the flow shop to the job shop. The best results are obtained by the two-job based approach.

Aggoune [Agg02, Agg04b] proposes a branch-and-bound method to the job-shop problem for makespan minimization. The approach can be generalized to any regular criterion. The disjunctive graph model is used for representing the nodes of trees. The author introduces an original way to take into account machines availability periods, introducing fictive jobs composed by unavailability periods and introduces flexibility on this latter. Finally, the calculation of lower bounds is based on the resolution of subproblems with two jobs, taking into account the precedence and availability constraints as well as latency dates on operations.

Resumable case

For the $J|rs(M_r^k)|C_{max}$ problem, Mauguière *et al.* [MBB03a] propose a branch-and-bound method. The computational results show that the problem with unavailability periods is a

more difficult than the one without unavailability periods.

Resumable/Non-resumable case

A branch-and-bound method is proposed by Mauguière *et al.* [MBB03b] to solve the $J, cr|rs/nrs(M_r^k)|C_{max}$ problem. In Mauguière *et al.* [MBB05], the authors extend the algorithms they developed for the job shop to solve the $J, cr/ncr|r_i, rs/nrs(M_r^k)|C_{max}$ problem.

3.1.6 Flexible job shop

Levitin [Lev00] investigates the multistage expansion problem for multistate series-parallel systems, where the objective is to minimize the sum of costs of the investments over the study period while satisfying reliability constraints at each stage. The stages form the study period; and at each stage the demand distribution is predicted in the form of a cumulative demand curve. The additional elements, chosen from a list of available products, and characterized by its capacity (productivity), availability, and cost, can be included into any system-component at any stage to increase the total system capacity and/or reliability. The author proposes a genetic algorithm where the solution encoding is integer strings representing multistage expansion planes. Reliability and cost estimations are the elements concerned by a solution quality.

Zribi [Zri05] develops a two-phase approach to solve separately the assignment and the sequencing problems. For the resolution of the assignment problem, a priority rule based heuristic is developed. The lower bound used is based on the relaxation of the non-preemption constraint and the use of Jackson's rule. Regarding the sequencing problem, a genetic algorithm is proposed. The temporized geometric approach, proposed by Aggoune [Agg02] is generalized.

Chan *et al.* [CWC06] develop a genetic algorithm-based approach to solve iteratively a resource-constrained operations-machines assignment problem and flexible job-shop scheduling problem. The flexibility introduced in the flexible shop floor can be quantified under different levels of resource availability.

Taghavi-Fard and Dehnar Saidy [TFDS09] develop an exact graphical algorithm based on the extension of Akers method for solving the problem related to the model $FJ, h_{kj}, cr/ncr|n = 2, rs/nrs/srs, r_i, m_j \leq 2|\gamma$, where γ is a performance measure based on the completion time.

The particularity of this algorithm is that it takes into account all the availability models for the flexible job shop environments, arbitrary number of resources (workcenters and processors), arbitrary holes on all work centers, and ready times.

3.1.7 Open shop

Non-preemptive case

Lu and Posner [LP93] develop a polynomial algorithm for the makespan minimization in the two-machine open shop problem, where one of the machines is not available at time zero, and the operations are non-preemptive.

Kubzin and Strusevich [KS06] consider the $O_2||C_{max}$ in which each machine has a maintenance activity, and whose duration depends on its starting time. The authors prove that the open shop problem is polynomially solvable for general functions defining the length of the maintenance periods.

Resumable case

Vairaktarakis and Sahni [VS95] show that for any number of machines and of unavailability periods, the problem of makespan minimization in the case of preemptive operations is a polynomial problem.

Breit [Bre00] studies the two-machine open shop problem with machine availability constraints. The author proves that there is no heuristics of performance guarantee for makespan minimization in case of resumable operations, if a machine has one unavailability period and the other two periods.

Breit *et al.* [BSS01b] address the makespan minimization for the two-machine open shop problem, where one of the machines has one unavailability period. The authors prove that the resumable problem is NP-hard in the weak sense and develop a heuristic with a ratio in the worst-case equal to $\frac{4}{3}$. In the case where a machine has several unavailability periods, a heuristic with an error of 2 is also developed.

Lorigeon *et al.* [LBB02b] study the $O_2|rs|C_{max}$ problem with a machine not always available. The problem is NP-hard. The authors propose a pseudo-polynomial time dynamic programming algorithm to optimally solve the problem when the machine is not available at time zero. A mixed integer linear program is suggested to optimally solve instances with up to 500 jobs in less than 5 min with CPLEX solver. It is proved that a worst-case error bound of any heuristic algorithm is equal to 1.

For the $O_2|rs|C_{max}$ problem, Kubzin *et al.* [KSBS05] present two polynomial-time approximation schemes: one for the problem with one unavailability period on each machine and the other for the problem with several unavailability periods on one of the machines. For problems with a more general structure of the unavailability intervals, if $P \neq NP$, there is no approximation scheme that is polynomial-time within a constant factor.

Non-resumable case

Breit *et al.* [BSS03] study the makespan minimization for the two-machine open shop problem in the non-resumable case. The authors assume first that one of the machines has several unavailability periods and show that there is no heuristic with performance guarantee. Then, a heuristic with worst-case ratio equal to 2 (resp. $\frac{4}{3}$) is developed for the case of one unavailability period on each machine (resp. on only one machine).

3.2 Problems with flexible availability constraints

3.2.1 Single machine

If the starting date of the unavailability is a decision variable, Qi *et al.* [QCT99] prove that the problem is NP-hard in the strong sense in case of several maintenance periods. The authors solve the non-resumable problem with a branch-and-bound method.

In case of semi-resumable operations, Graves and Lee [GL99] consider unavailability periods due to preventive maintenance; and are interested in the study of two optimization criteria: the weighted sum of jobs completion dates and the maximum of delays. The starting dates of maintenance activities are decision variables. Thus, two scenarios on the production horizon are assumed. In case of a too long horizon as compared to the maintenance period, the

problem is NP-hard. The authors develop a pseudo-polynomial algorithm based on dynamic programming. However, in case of a quite short horizon, it is sometimes impossible to continue the operation of maintenance; it must therefore finish in the next horizon. This scenario is also NP-hard. Nonetheless, SPT (resp. EDD) rules allowed solving the problem in an exact manner in the case of the minimization of sum of completion dates (resp. minimizing of maximum of advances).

Li and Cao. [LC95] study the same problem. The optimization criterion is the maximum of delays. A branch-and-bound algorithm is proposed for small instances. Larger instances are solved with the EDD rule based heuristic.

Kaabi [Kaa04] is interested in production and maintenance scheduling problem on a machine. The maintenance activities are assumed flexible (periodic), the optimization criterion that is considered takes into account both aspects of the production and the maintenance. Different heuristics are introduced to solve the problem.

3.2.2 Flow shop

Aggoune [Agg02] studies the flow shop problem with non-preemptive operations and flexible unavailability periods. The author assumes that a time window is allocated to each unavailability period. In the solution method, each unavailability period is shifted to the right in its time window. After scheduling all the jobs, the unavailability periods are moved to the left, when possible, to reduce idle times on the machines. Then, the operations succeeding to the unavailability period are moved to the left depending on the completion dates of the operations preceding them in the job routings.

Kaabi [Kaa04] is interested in the study of the flow shop in presence of periodic maintenance periods. The author proposed a branch-and-bound method and a genetic algorithm for solving the problem.

3.2.3 Job shop

Aggoune [Agg02] tackles the job shop problem with non-preemptive operations using the 2-job geometric approach. The moving of the unavailability periods is performed before all the jobs

of the treatment sequence are scheduled. Hence, an unavailability period succeeding to an idle time on a machine is moved to the right if no operation remaining to schedule on the machine can be inserted in that idle time.

Harrath [Har03] is interested in the job shop problem with machines periodic maintenance activities trying to optimize two criteria: the makespan and the sum of advances and delays costs of maintenance. For the resolution, the author develops a multi-objective genetic algorithm.

Zribi [Zri05] develops heuristics based on a sequential strategy to study the problem with non-preemptive operations. The flexibility on starting dates of unavailability periods is treated as follows:

First heuristic: The unavailability periods are placed totally in the right in their time windows. Each time that an operation has to be scheduled on the machine, it tests if it can be processed before of after the next unavailability period of the processing machine. In case where it must be inserted after that unavailability period, this unavailability period is moved as early as possible in its time window to reduce the idle time.

Second heuristic: For each machine, all possible positions of the unavailability periods are tested. Indeed, an unavailability period can be inserted in all the positions of its time window. Two positions are possible: at the end of each operation that is in the interval, or at the starting date of the interval if their is no operation that is being processed at that time. The procedure used to insert the unavailability period is a modification of a branch-and-bound used by Benbouzid [Ben05] to solve the scheduling of production and maintenance in a flow shop with permutation.

Third heuristic: To calculate the due date of each operation, the unavailability periods are placed totally at the right of their time windows. To each operation is associated a due date which is equal to the starting date of the following operation in the job in this schedule which duration represents an upper bound of the problem. The problem is equivalent to a job shop problem with deadlines. EDD (Earliest Due Date) priority rule is used to schedule simultane-

ously the production and the maintenance.

3.2.4 Flexible job shop

Gao *et al.* [GGS06] investigate the flexible job shop scheduling problem with non-preemptive operations and non-fixed unavailability periods on machines: the completion time of a maintenance period is not fixed and is determined during the schedule construction. To solve the problem, the authors propose a hybrid genetic algorithm. Two kinds of neighborhoods, based on the concept of critical path, are proposed. A local search procedure is added to the genetic algorithm.

3.3 Conclusion

A state-of-the art covering the production scheduling problems including resources unavailability periods was presented in this Chapter. At first, problems with fixed machine unavailability periods were addressed. Then, we have dealt with the treatment, in literature, of the flexibility on unavailability periods.

Although research efforts were deployed to machine scheduling problems integrating resource availability constraints, they are essentially concentrated on problems with fixed availability periods. Moreover, the most studied problems are: single machine, parallel machines and flow shop problems.

Part II

Mathematical modeling and Resolution methods

Chapter 4

Mathematical modeling

We present in this chapter a mathematical modeling approach to tackle the job shop scheduling problem with resource availability constraints. This approach is often neglected by researchers for the strong complexity of the problem. We choose to develop this approach as it allows to solve some problem sizes, how to deal with resource unavailability constraints, to prove the relevance of introducing flexibility to the problem and evaluate the quality of solutions provided by the approximation methods and the column generation approach we develop.

We recall that the machines unavailability periods are known in advance. We introduce flexibility on starting dates of machine unavailability periods, i.e. an unavailability period can start in a time window defined by earliest and latest starting dates (Aggoune [Agg02, Agg04]). Moving the unavailability period in its time window allows the creation of an idle time on the resource to process a job operation earlier. We also introduce flexibility on durations of the unavailability periods. Depending on management decisions, a priority can be given to production. Indeed, it is possible when elaborating a schedule to modify durations of unavailability periods if, for each machine, the minimum number of unavailability periods and the minimum total unavailability duration are satisfied. We deal with the preemption between job operations and machine unavailability periods, as it seems relevant to study the possibility for an operation to be interrupted or not to better model the reality of industry. When preemption is allowed, a job operation can be interrupted by a machine unavailability period and then resumed, possibly incurring a penalty, as soon as the resource is available again. This can be the case of products that are grouped in lots. We then assume that, if an operation can be interrupted, it is only due to a machine unavailability period. Thus, we study the following

different cases: strictly non-preemptive, resumable, non-resumable, semi-resumable operations and crossable and non-crossable unavailability periods. We are interested at first in minimizing the makespan.

Models of the job shop problem including the availability models are presented and extended to consider other optimization criteria, constraints on jobs and to model the flexible job shop scheduling problem with machine unavailability.

The chapter is organized as follows: in Section 4.1, the job shop scheduling problem with resource availability constraints is presented and models are given. Extensions of these models are discussed in Section 4.2.

4.1 Job shop problem with limited resource availability

We assume at first that preemption is not allowed (an operation cannot be interrupted by another operation or an unavailability period). A Mixed-Integer Linear Programming and an Integer Linear Programming models are presented and compared. The first model is using disjunctive variables, and the second model time-indexed variables. A mathematical model is then proposed which integrates the possibility for an operation to be interrupted or not and the possibility for an unavailability period to interrupt an operation or not. The model requires coefficients for operations representing penalties on preemption and for unavailability periods representing possibilities of preemption. To our knowledge, there is currently no study which includes all cases and, according to assumptions on operations or unavailability periods, new problems are considered. This generalization enables us to combine all problems into one and evaluate the relevance of some ideas that make sense but were not proved in literature yet.

This part is organized as follows. Section 4.1.1 presents mathematical models for the problem in the non-preemptive case and discusses the test results. Section 4.1.2 introduces the general disjunctive model and presents test results. All these tests were performed on generated benchmarks with a standard solver: ILOG CPLEX 10. Section 4.1.3 presents the numbers of variables and constraints induced by the models.

4.1.1 Mathematical models for the non-preemptive problem

Two mathematical models are presented and compared as scheduling problems can be modeled in two usual ways as integer linear programs: by using disjunctive variables or time-indexed variables. The first model is based on the disjunctive graph of Roy and Susmann [RS64] for

the job shop scheduling problem. The second one is using time-indexed variables and is based on the formulation of Pritsker *et al.* [PWW69] for the resource constrained project scheduling problem (RCPSP). In the latter, we need to introduce a schedule length and time windows also for starting dates of job operations.

The disjunctive formulation is a very natural way to model the problem. The relations between operations and unavailability periods are easy to express as inequalities, and less variables and constraints are necessary than the second model. But the Linear Programming relaxations (i.e. dropping the integrality requirements on the variables) of the time-indexed formulation provide strong bounds, usually better than the bounds provided by other mixed integer programming models (Van Den Akker *et al.* [AHS00] for the single-machine scheduling problem and Demassey *et al.* [DAM03] for the RCPSP). Moreover, many constraints such as deadlines and release dates can simply be handled by fixing some variables to 0 (Van Den Akker *et al.* [AHS00]). However, the number of variables and constraints induced by the time-indexed formulation can be huge depending on the numbers of jobs, machines, processing times of operations and durations of unavailability periods.

The disjunctive formulation is presented in Section 4.1.1.1 and the time-indexed formulation in Section 4.1.1.2. Section 4.1.1.3 presents and discusses test results performed on ILOG CPLEX 10.

4.1.1.1 The disjunctive formulation

This formulation is deduced from the one by Applegate and Cook [AC91] based on the mathematical formalism of the disjunctive graph of Roy and Susman [RS64]. Let $G = (N, A, E)$ denote a graph, where the set N of nodes is formed of job operations and machine unavailability periods plus the dummy start and finish operations (which processing times are equal to 0). We use Aggoune's idea [Agg02, Agg04], i.e. each machine can be seen as a job whose operations are the unavailability periods. A is the set of conjunctive arcs between every two consecutive operations on a routing, between every two consecutive unavailability periods on a machine, and between the dummy start (resp. finish) operation and the first (resp. last) operation of each job or unavailability period of each machine. E is the set of disjunctive arcs that link operations of different jobs processed on the same machine, and link operations and unavailability periods on the same machine. The advantage of the formulation is that disjunctive constraints are easy to write. Note that a conjunctive arc leads to one constraint whereas a disjunctive arc corresponds to the choice between two constraints. Indeed, each disjunctive arc consists of a

pair of arcs with opposite orientations such that any path through the graph contains only one of them.

This approach based on the disjunctive graph is also used by Gao *et al.* [GGS06] to model the flexible job shop scheduling problem with resource availability constraints for the non-preemptive case.

This model is studied according to three assumptions. First, we suppose that the starting dates of the unavailability periods are fixed and we try to solve the resulting model. Then, we assume that the starting dates of each unavailability period are not fixed but vary within a time window. Finally, we present the case of the problem without unavailability period.

Fixed starting dates for unavailability periods

The following additional variables are introduced:

$X_{ij,i'j'}$: Binary variable to define which of operations O_{ij} and $O_{i'j'}$ is processed before the other. It is equal to 1 if O_{ij} is scheduled before $O_{i'j'}$ and 0 otherwise,

$Y_{ij,rk}$: Binary variable to define which of operation O_{ij} and unavailability period h_{rk} starts before the other. It is equal to 1 if O_{ij} starts before h_{rk} and 0 otherwise.

The model is as follows:

$$MinC_{max} \tag{4.1}$$

$$t_{i(j+1)} \geq t_{ij} + p_{ij} \qquad i = 1,..,n; j = 1,..,n_i - 1 \tag{4.2}$$

$$t_{ij} - t_{i'j'} + MX_{ij,i'j'} \geq p_{i'j'} \qquad \forall O_{ij}, O_{i'j'}\ (O_{ij} \neq O_{i'j'})\ s.t.\ mr_{ij} = mr_{i'j'} \tag{4.3}$$

$$t_{i'j'} - t_{ij} + M(1 - X_{ij,i'j'}) \geq p_{ij} \qquad \forall O_{ij}, O_{i'j'}\ (O_{ij} \neq O_{i'j'})\ s.t.\ mr_{ij} = mr_{i'j'} \tag{4.4}$$

$$t_{ij} - S_{rk} + MY_{ij,rk} \geq p'_{rk} \qquad \forall h_{rk}, \forall O_{ij}\ s.t.\ mr_{ij} = M_r \tag{4.5}$$

$$S_{rk} - t_{ij} + M(1 - Y_{ij,rk}) \geq p_{ij} \qquad \forall h_{rk}, \forall O_{ij}\ s.t.\ mr_{ij} = M_r \tag{4.6}$$

$$Y_{ij,r(k+1)} - Y_{ij,rk} \geq 0 \qquad \forall h_{rk}, \forall O_{ij}\ s.t.\ mr_{ij} = M_r \tag{4.7}$$

$$C_{max} \geq t_{in_i} + p_{in_i} \qquad i = 1,..,n \tag{4.8}$$

$$C_{max} \geq 0 \tag{4.9}$$

$$t_{ij} \geq 0 \qquad i = 1,..,n; j = 1,..,n_i \tag{4.10}$$

$$X_{ij,i'j'} \in \{0,1\} \qquad \forall O_{ij}, O_{i'j'}\ (O_{ij} \neq O_{i'j'})\ s.t.\ mr_{ij} = mr_{i'j'} \tag{4.11}$$

$$Y_{ij,rk} \in \{0,1\} \qquad \forall h_{rk}, \forall O_{ij}\ s.t.\ mr_{ij} = M_r \tag{4.12}$$

where M is a very large value.

The objective function [4.1] is the minimization of the makespan C_{max}. The conjunctive constraint [4.2] ensures that an operation $O_{i'j'}$ which follows an operation O_{ij} in the routing cannot start before the completion of O_{ij}. The disjunctive constraints [4.3] and [4.4] guarantee

the non overlapping of operations O_{ij} and $O_{i'j'}$ that must be processed on the same machine, i.e. that operation O_{ij} is either processed before or after operation $O_{i'j'}$. Similarly, Constraints [4.5] and [4.6] ensure that operation O_{ij} is either processed before or after unavailability period h_{rk}. Disjunctive constraints are associated to resource constraints which express the fact that, when a machine is available, it can process only one operation at a time. Constraint [4.7] implies that, if operation O_{ij} precedes unavailability period h_{rk}, then it precedes all unavailability periods which follow h_{rk}. Moreover, if operation O_{ij} follows unavailability period $h_{r(k+1)}$, then it follows all unavailability periods which precede $h_{r(k+1)}$. Constraints [4.8] indicates that the schedule cannot be completed before the end of the last operation of each job. Constraints from [4.9] to [4.12] provide bounds on the variables of the model.

Non-fixed starting dates for unavailability periods

Here we assume that, in our problem, the starting date S_{rk} of unavailability period h_{rk} can vary in the interval $[ES_{rk}, LS_{rk}]$. Then the following constraints are added to the model in case of fixed starting dates of unavailability periods:

$$S_{rk} \geq ES_{rk} \qquad \forall h_{rk} \tag{4.13}$$

$$S_{rk} \leq LS_{rk} \qquad \forall h_{rk} \tag{4.14}$$

Constraint [4.13] (respectively, [4.14]) ensures that S_{rk} starts after its earliest starting date ES_{rk} (respectively, before its latest starting date LS_{rk}).

Without unavailability periods

Referring to the model in the case of fixed starting dates for unavailability periods, when the machines are continuously available, disjunctive constraints [4.4] and [4.5] between the operations and the unavailability periods, transitivity constraint [4.7] between unavailability periods on the same machine, and variables $Y_{ij,rk}$ are removed.

4.1.1.2 Time-indexed formulation

This formulation, which to our knowledge has not been proposed before, is based on the one by Pritsker *et al.* [PWW69] for the resource constrained project scheduling problem. The differences are that in the formulation of Pritsker *et al.* [PWW69], a job has only one operation, there are no unavailability periods on resources, a resource can be used to process at least one operation, and the variables model the completion dates of the operations.

Problem with non-fixed starting dates for unavailability periods

In the formulation below, we need to know the schedule length T, unavailability period h_{rk} must start in the interval $[ES_{rk}, LS_{rk}]$ and also the processing of each operation O_{ij} must start

in the time window defined by the interval $[Et_{ij}, Lt_{ij}]$ where Et_{ij} (resp. Lt_{ij}) is the earliest (resp. latest) starting date of O_{ij}. The interval is relevant to reduce the number of variables.

The variables of the model are:

x_{ij}^t: Binary variable which is equal to 1 if operation O_{ij} starts at time t and 0 otherwise,

y_{rk}^t: Binary variable which is equal to 1 if unavailability period h_{rk} starts at time t and 0 otherwise.

Note that the same variable can be used for both operations and unavailability periods.

The starting dates of O_{ij} and h_{rk} are respectively $\sum_{t=Et_{ij}}^{Lt_{ij}} t x_{ij}^t$ and $\sum_{t=ES_{rk}}^{LS_{rk}} t y_{rk}^t$.

The model is as follows:

$$Min C_{max} \tag{4.15}$$

$$\sum_{t=Et_{ij}}^{Lt_{ij}} x_{ij}^t = 1 \qquad i = 1, .., n; \quad j = 1, .., n_i \tag{4.16}$$

$$\sum_{t=ES_{rk}}^{LS_{rk}} y_{rk}^t = 1 \qquad r = 1, .., m; \quad k = 1, .., m_r \tag{4.17}$$

$$\sum_{t=Et_{i(j+1)}}^{Lt_{i(j+1)}} t x_{i(j+1)}^t - \sum_{t=Et_{ij}}^{Lt_{ij}} t x_{ij}^t \geq p_{ij} \qquad i = 1, .., n; \quad j = 1, .., n_i - 1 \tag{4.18}$$

$$C_{max} - \sum_{t=Et_{in_i}}^{Lt_{in_i}} t x_{in_i}^t \geq p_{in_i} \qquad i = 1, .., n \tag{4.19}$$

$$\sum_{i=1}^{n} \sum_{j=1; mr_{ij}=r}^{n_i} \sum_{q=max[t-p_{ij}+1, Et_{ij}]}^{min[t, Lt_{ij}]} x_{ij}^q + \sum_{k=1}^{m_r} \sum_{q=max[t-p_{rk}+1, ES_{rk}]}^{min[t, LS_{rk}]} y_{rk}^q \leq 1 \qquad t = 0, .., T-1; \quad r = 1, .., m \tag{4.20}$$

$$C_{max} \geq 0 \tag{4.21}$$

$$x_{ij}^t \in \{0, 1\} \qquad i = 1, .., n; j = 1, .., n_i; \quad t = Et_{ij}, .., Lt_{ij} \tag{4.22}$$

$$y_{rk}^t \in \{0, 1\} \qquad r = 1, .., m; k = 1, .., m_r; \quad t = ES_{rk}..LS_{rk} \tag{4.23}$$

The objective function [4.15] is the minimization of the makespan. Constraint [4.16] (resp. [4.17]) expresses the fact that an operation O_{ij} (resp. an unavailability period h_{rk}) must be processed once. Constraint [4.2] is replaced by Constraint [4.18] to ensure the precedence constraints between start dates of consecutive operations in job routings. Constraint [4.19] replaces Constraint [4.8] and ensures that the schedule cannot end before the completion of the last operation of each job. Precedence constraints between operations of different jobs or between operations and unavailability periods which must be scheduled on the same machine are implicitly expressed through Constraint [4.20]. Indeed, at most only one operation can be processed at time t but only if the machine is available. This is the resource constraint. It

replaces the disjunctive constraints ([4.3] to [4.6]). Constraints [4.21] to [4.23] provide bounds on the variables of the model.

If schedule length T and time windows of O_{ij} and h_{rk} are large, the formulation contains a huge number of variables and resource constraints.

Machines without unavailability periods

From the previous model, Constraint [4.17] and variables y_{rk}^t are removed and Constraint [4.20] is modified by deleting the term $\sum_{k=1}^{m_r} \sum_{q=max[t-p_{rk}+1,ES_{rk}]}^{min[t,LS_{rk}]} y_{rk}^q$.

Another way to consider unavailability periods without modifying the model is to define fictitious unavailability periods at the end of the schedule length such as it has no influence; but being sure that this value of the schedule length is not reached.

4.1.1.3 Numerical results

Numerical experiments have been performed using a standard solver, ILOG CPLEX release 10, on some benchmarks which are generated following a procedure proposed by Aggoune [Agg02]. Four classes of five benchmarks were generated, where the pairs (number of machines, number of jobs) are as follows: $(5, 5)$, $(5, 10)$, $(10, 10)$, $(10, 15)$. Each job visits all the machines exactly once. The processing time of each operation is randomly generated from the set of values $\{50, 60, .., 140, 150\}$.

Each machine has two unavailability periods. Their positions are generated so that they will have enough influence. The position of the first unavailability period is randomly generated, and the position of the second one is generated so that the distance between the two unavailability periods is at least equal to the longest processing time of the operations on the machine to which are added the lengths of the intervals $[S_{r1}, LS_{r1}]$, $[ES_{r2}, S_{r2}]$, which are set to 20. This is due to the fact that, for the general disjunctive model, if two unavailability periods are too close and if an operation is interrupted by the first unavailability period, it can overlap with the second one. The duration of an unavailability period on a machine is chosen as the average of the processing times of the operations to be processed on this machine.

The resolution time limit for each benchmark was set to 60 minutes.

Table 4.1 summarizes the test results for the disjunctive formulation when the machines are continuously available, and when the starting dates of unavailability periods are fixed and non-fixed. The first column is the name of the benchmark which is of type $XmYnZ$, where X, Y and Z are respectively the number of machines, number of jobs and of the benchmark number in the class. Columns 2 to 5 give the results for the mixed integer linear program when no unavailability periods are planned on the machines. Columns 6 to 9 show the results for the mixed integer linear program when the starting dates of the unavailability periods are fixed, and Columns 10 to 13 show the results for the case of non-fixed dates. Columns 2, 6 and 10 correspond to the best lower bound, and Columns 3, 7 and 11 to the objective function of the best solution. Columns 4, 8 and 12 show the gap (expressed in percentage), i.e.

$$gap = \frac{|\text{Best solution} - \text{Best bound}|}{|\text{Best solution}|}$$

Columns 5, 9 and 13 give the CPU time (in seconds).

ILOG CPLEX solved optimally benchmarks up to 10 machines and 10 jobs and provided feasible solutions to benchmarks of the (10,15) class. The results show that adding unavailability periods add more complexity to the problem. Indeed, it is more time consuming and only a feasible solution was provided to the instances 5m10n2 and 5m10n3 that were solved to optimality when the machines are continuously available. Moreover, It can be useful to allow starting dates of unavailability periods to vary within some time windows. It provides better values for the objective function. Indeed, for each unavailability period, the best slot is defined in its time window to create an idle time on the machine before or after to process an operation as early as possible. However, introducing the flexibility has a cost: it is more time consuming to solve the model for most of the benchmarks.

For the time-indexed formulation, the upper bounds on the horizon length T we assumed for Classes $(5, 5)$, $(5, 10)$ and $(10, 10)$ are respectively 1500, 2000, 2500. These values were chosen based on the optimal solutions or best feasible solutions provided by CPLEX for the disjunctive formulation. These values are not very close or very far from the objective values of the solutions in order not to advantage or disadvantage too much the time-indexed formulation.

The earliest starting date of an operation O_{ij} is obtained by adding its processing time to the earliest starting date of the previous operation in the job routing, starting from 0. The latest starting date of an operation O_{ij} is obtained by subtracting its processing time from the latest starting date of the operation which immediately succeeds O_{ij} in the job routing, starting from T.

Tables 4.2, 4.4 and 4.6 provide the average numbers of variables and constraints per class of benchmarks. Column 1 is the benchmark class. Columns 2 and 3 give respectively the number of machines and the number of jobs in the class. Columns 4 and 5 show respectively the number of variables and constraints for the disjunctive formulation with non-fixed starting dates of unavailability periods, whereas Columns 6 and 7 present the same numbers for the time-indexed formulations.

The test results comparing the resolution of the disjunctive formulation and the time-indexed formulation are gathered in Tables 4.3, 4.5 and 4.7. Only results for the classes up to (10,10) are given. Note that the names of the benchmarks are modified by adding respectively div10 and div20 to the end of the initial name of the benchmark when the data are respectively divided by 10 and 20.

The test results for initial benchmarks are summarized in Table 4.3. Columns 2 to 7 present the results of the disjunctive formulation, whereas Columns 8 to 13 present the results of the time-indexed formulation. Columns 2 and 8 (resp. 3 and 9) give the linear relaxation (resp. the CPU time) of the formulations. Columns 4 to 7 present the results of the disjunctive mixed integer program, whereas Columns 10 to 13 present the results of the time-indexed integer linear program. Columns 4 and 10 correspond to the best lower bound, and Columns 5 and 11

Table 4.1: Test results for non-preemptive disjunctive formulation.

Problem	Without unavailability periods				Fixed unavailability periods				Non-fixed unavailability periods			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	660	660	0	0.01	895	895	0	0.11	825	825	0	0.07
5m5n2	770	770	0	0.01	1096	1096	0	0.02	1076	1076	0	0.02
5m5n3	790	790	0	0.01	1070	1070	0	0.19	1034	1034	0	0.22
5m5n4	910	910	0	0.01	1147	1147	0	0.16	1108	1108	0	0.28
5m5n5	890	890	0	0.04	1202	1202	0	0.25	1182	1182	0	0.63
5m10n1	1080	1080	0	22.06	1361	1361	0	56.98	1300	1300	0	51.58
5m10n2	1180	1180	0	257.51	1435	1455	1.37	3595.84	1278	1400	8.71	3580.74
5m10n3	1270	1270	0	90.17	1497	1607	6.85	3596.27	1427	1578	9.57	3590.55
5m10n4	1260	1260	0	20.12	1537	1537	0	998.22	1492	1492	0	2555.96
5m10n5	1130	1130	0	25.87	1415	1415	0	39.6	1372	1372	0	255.59
10m10n1	1650	1650	0	120.70	1948	1948	0	432.25	1824	1847	1.25	3589.91
10m10n2	1640	1640	0	206.15	1917	1917	0	204.18	1839	1839	0	1303.69
10m10n3	1520	1520	0	64.01	1875	1875	0	2662.98	1773	1773	0	836.54
10m10n4	1500	1500	0	305.73	1772	1772	0	142.57	1693	1693	0	3591.18
10m10n5	1590	1590	0	333.62	1933	1933	0	529.31	1833	1833	0	3477.83
10m15n1	1578.46	1820	13.27	3600	1723	2129	19.07	3577.96	1587	2116	25	3569.84
10m15n2	1480.00	1950	24.10	3600	1838	2335	21.28	3521.38	1729	2288	24.43	3574.02
10m15n3	1431.50	1910	25.05	3600	1721	2241	23.20	3525.25	1637	2167	24.46	3577.09
10m15n4	1460.00	1900	23.16	3600	1744	2204	20.87	3523.81	1526	2167	29.58	3559.17
10m15n5	1476.04	1990	25.83	3600	1758	2339	24.84	3541.55	1619	2147	24.59	3560.27

to the objective function of the best solution. Columns 6 and 12 give the gap whereas Columns 7 and 13 give the CPU time (in seconds).

The test results for modified benchmarks are presented in Table 4.5 and 4.7. Columns 2 to 5 present the results of the disjunctive mixed integer program, whereas Columns 6 to 9 present the results of the time-indexed integer linear program. Columns 2 and 6 correspond to the best lower bound, and Columns 3 and 7 to the objective function of the best solution. Columns 4 and 8 show the gap. Columns 5 and 9 give the CPU time (in seconds).

Table 4.2: Number of variables and constraints for initial benchmarks for non-preemptive formulations.

Problem class	m	n	Disjunctive formulation		Time-indexed formulation	
			Number of variables	Number of constraints	Number of variables	Number of constraints
(5,5)	5	5	136	250	25446	7565
(5,10)	5	10	386	750	75331	10115
(10,10)	10	10	771	1500	149960	25230

Table 4.2 shows that the time-indexed formulation induces a huge number of variables and constraints compared to the disjunctive one. From Table 4.3, it can be observed that the time-indexed formulation give better linear relaxation than the disjunctive formulation but with a larger CPU time. It can also be observed that, due to the huge number of variables and constraints, the test results of the time-indexed formulation are bad compared to the test results of the disjunctive formulation. Indeed, even for small benchmarks (Class (5,5)) which were optimally solved in less than one second for the disjunctive formulation, only feasible solutions are obtained for the time-indexed formulation except for one benchmark that is solved to optimality. For larger benchmarks, the latter was not able to provide a feasible solution after 60 minutes, whereas it is possible to solve benchmarks up to 10 machines and 10 jobs using the disjunctive formulation.

To reduce the numbers of variables and constraints for the time-indexed formulation, we modified benchmarks by dividing by 10 the schedule length and the durations and the dates of operations and unavailability periods. We ensured to get integer values by rounding the non-integer values to the immediately upper integer value. This implies rounded values of the objective function for some solutions.

Table 4.4 shows that the modification of the benchmark data decreases the number of variables and constraints for the time-indexed formulation by approximatively 10 times whereas it has no influence for the disjunctive formulation.

From Table 4.5, it can be observed that, for the time-indexed formulation, problems with 5 machines and 5 jobs are optimally solved and feasible solutions are obtained for problems with 5 machines and 10 jobs. However, we still obtain better results with the disjunctive formulation. It seems then more relevant to model the problem using the disjunctive formulation. However, the results of the time-indexed formulation can be improved by elaborating an appropriate

Table 4.3: Test results for initial benchmarks for non-preemptive formulations.

Problem	Disjunctive formulation						Time-indexed formulation					
	Linear relaxation	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)	Linear relaxation	CPU	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	550	0	**825**	**825**	0	0.07	725.62	1.15	793.89	915	13.24	2058.89
5m5n2	610	0	**1076**	**1076**	0	0.02	801.49	2.09	**1076.00**	**1076**	0	102.85
5m5n3	600	0	**1034**	**1034**	0	0.22	722	1.07	789.36	1486	46.88	1925.21
5m5n4	650	0	**1108**	**1108**	0	0.28	853.85	1.25	921.16	1239	25.65	3208.93
5m5n5	590	0	**1182**	**1182**	0	0.63	841.11	2.48	889.55	1255	29.12	2978.15
5m10n1	630	0.01	**1300**	**1300**	0	51.58	943.28	124.77	u			907.68
5m10n2	610	0.01	1278	**1400**	8.71	3580.74	956.49	161.21	u			908.36
5m10n3	600	0	1427	**1578**	9.57	3590.55	1078.02	141.74	u			908.96
5m10n4	650	0.01	**1492**	**1492**	0	2555.96	1084.84	57.95	u			909.22
5m10n5	630	0	**1372**	**1372**	0	255.59	905.82	39.33	u			909.38
10m10n1	1170	0.01	1824	**1847**	1.25	3589.91	1451.70	83.24	u			916.79
10m10n2	1200	0.01	**1839**	**1839**	0	1303.69	1373.89	182.15	u			916.44
10m10n3	1130	0.01	**1773**	**1773**	0	836.54	1388.47	153.55	u			916.90
10m10n4	1170	0.01	**1693**	**1693**	0	3591.18	1346	18.49	u			915.05
10m10n5	1140	0.01	**1833**	**1833**	0	3477.83	1351.85	87.22	u			914.00

bold: best solution

u: unknown

heuristic to calculate better earliest and latest starting dates of the operations to reduce the number of variables. Note that we tested several modifications of CPLEX parameters to improve the results.

Table 4.4: Number of variables and constraints for modified benchmarks for non-preemptive formulations by scale $\frac{1}{10}$.

Problem class	m	n	Disjunctive formulation		Time-indexed formulation	
			Number of variables	Number of constraints	Number of variables	Number of constraints
(5,5)	5	5	136	250	2577	815
(5,10)	5	10	386	750	7582	1115
(10,10)	10	10	771	1500	15105	2730

Table 4.5: Test results for modified benchmarks for non-preemptive formulations by scale $\frac{1}{10}$.

Problem	Disjunctive formulation				Time-indexed formulation			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div10	**83**	**83**	0	0.08	**83**	**83**	0	19.94
5m5n2div10	**103**	**103**	0	0.05	**103**	**103**	0	9.14
5m5n3div10	**104**	**104**	0	0.05	**104**	**104**	0	28.19
5m5n4div10	**111**	**111**	0	0.17	**111**	**111**	0	57.91
5m5n5div10	**119**	**119**	0	0.45	**119**	**119**	0	778.29
5m10n1div10	**130**	**130**	0	509.04	102.19	177	42.26	3302.59
5m10n2div10	138	**140**	1.43	3598.07	96	159	39.62	3442.16
5m10n3div10	137	**158**	13.29	3532.43	119	195	38.97	3294.46
5m10n4div10	135	**149**	9.40	3598.03	120	168	28.57	2867.04
5m10n5div10	135	**137**	1.46	3598.33	105	166	36.75	2605.20
10m10n1div10	**185**	**185**	0	971.87	u			1702.90
10m10n2div10	**184**	**184**	0	831.94	u			2511.23
10m10n3div10	**178**	**178**	0	1474.63	u			2148.58
10m10n4div10	**169**	**169**	0	478.14	u			3163.02
10m10n5div10	177	**184**	3.80	3594.17	u			926.02

bold: best solution

u: unknown

To improve again the results of the time-indexed formulation, the initial benchmarks were modified by dividing by 20 the schedule length, the processing times and the durations of the operations and the unavailability periods.

Table 4.6 shows that the number of variables and constraints for the time-indexed formulation decreases considerably whereas it has no influence for the disjunctive formulation. The results of the resolution of the instances by the time-indexed formulation are improved again (see Table 4.7).

Table 4.6: Number of variables and constraints for modified benchmarks for non-preemptive formulations by scale $\frac{1}{20}$.

Problem class	m	n	Disjunctive formulation		Time-indexed formulation	
			Number of variables	Number of constraints	Number of variables	Number of constraints
(5,5)	5	5	136	250	1635	440
(5,10)	5	10	386	750	4190	615
(10,10)	10	10	771	1500	10595	1730

Table 4.7: Test results for modified benchmarks for non-preemptive formulations by scale $\frac{1}{20}$.

Problem	Disjunctive formulation				Time-indexed formulation			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div20	**34**	**34**	0	0.06	**34**	**34**	0	0.65
5m5n2div20	**40**	**40**	0	0.06	**40**	**40**	0	1.54
5m5n3div20	**42**	**42**	0	0.07	**42**	**42**	0	26.93
5m5n4div20	**47**	**47**	0	0.07	**47**	**47**	0	1
5m5n5div20	**47**	**47**	0	0.10	**47**	**47**	0	37.15
5m10n1div20	50.54	**59**	14.33	3600	46	70	2.13	3600
5m10n2div20	**66**	**66**	0	1414.94	60	**66**	3.06	3600
5m10n3div20	57	**73**	21.92	3600	75	78	30.85	3600
5m10n4div20	62	**72**	13.89	3600	68	78	12.82	3600
5m10n5div20	60	**64**	6.25	3600	55	71	22.54	3600
10m10n1div20	79	**90**	12.22	3600	u			3600
10m10n2div20	**87**	**87**	0	858.98	u			3600
10m10n3div20	**81**	**81**	0	343.98	u			3600
10m10n4div20	73	**82**	10.98	3600	u			3600
10m10n5div20	**84**	**84**	0	1760.63	69	150	54	3600

bold: best solution

u: unknown

4.1.2 General disjunctive model

As the disjunctive formulation provides better results for the non-preemptive problem than the time-indexed model, we generalize it to include all problems dealt in the literature (see Section 4.1.1). To our knowledge, there is currently no study which includes all cases and, depending on the assumptions on operations or unavailability periods, new problems are considered. This generalization enables us to combine all problems into one. Coefficients of penalty and preemption introduced respectively for operations and unavailability periods allow each of them to have its own characteristic; this helps us to model a large number of workshop configurations. Note that there is a disjunction between an operation and an unavailability period if the latter starts before the operation or if the unavailability period is non-crossable or the operation is non-preemptive. The case where the operation is semi-resumable and the unavailability period is crossable can be easily deduced from the disjunctive constraints.

The model is presented in Section 4.1.2.1. In Section 4.1.2.2 we introduce flexibility on the unavailability periods. Finally, Section 4.1.2.3 presents and discusses test results performed on the generated benchmarks with ILOG CPLEX 10.

4.1.2.1 Mathematical model

Note that the completion date of an operation O_{ij} cannot be completely represented by $t_{ij}+p_{ij}$ when O_{ij} is interrupted by unavailability period h_{rk}, since this quantity will be increased by the duration $p_{rk}^{'}$ of the unavailability period and the proportion of the operation to redo and which spans from 0 to p_{ij}. It is thus necessary to use the completion date C_{ij} as a variable of the model to model this case, otherwise the durations will overlap.

The model uses the coefficients of penalty on preemption α_{ij} and the coefficient of preemption β_{ijk}

The following additional variable is introduced:

$Z_{ij,rk}$: Binary variable which is equal to 1 if operation O_{ij} starts before unavailability period h_{rk} (i.e. $Y_{ij,rk}$ is equal to 1) and finishes after h_{rk}, and is equal to 0 otherwise.

The model is as follows:

$$\min C_{max} \tag{4.24}$$

$$t_{i(j+1)} \geq C_{ij} \qquad i = 1,..,n; j = 1,..,n_i - 1 \tag{4.25}$$

$$t_{ij} + MX_{ij,i'j'} \geq C_{i'j'} \qquad \forall\ (O_{ij} \neq O_{i'j'}) \text{ s.t. } mr_{ij} = mr_{i'j'} \tag{4.26}$$

$$t_{i'j'} + M(1 - X_{ij,i'j'}) \geq C_{ij} \qquad \forall\ (O_{ij} \neq O_{i'j'}) \text{ s.t. } mr_{ij} = mr_{i'j'} \tag{4.27}$$

$$t_{ij} - S_{rk} + MY_{ij,rk} \geq p_{rk}^{'} \qquad \forall h_{rk}, \forall O_{ij} \text{ s.t. } mr_{ij} = M_r \tag{4.28}$$

$$-t_{ij} + S_{rk} + M(1 - Y_{ij,rk}) \geq (1 - \beta_{ijk}Z_{ij,rk})p_{ij} + \varepsilon\beta_{ijk}Z_{ij,rk} \qquad \forall h_{rk}, \forall O_{ij} \text{ s.t. } mr_{ij} = M_r \tag{4.29}$$

$$\beta_{ijk}Z_{ij,rk} \leq \beta_{ijk}Y_{ij,rk} \qquad \forall h_{rk}, \forall O_{ij} \text{ s.t. } mr_{ij} = M_r \tag{4.30}$$

$$C_{ij} - t_{ij} + \beta_{ijk}MZ_{ij,rk} \geq p_{ij} \qquad \forall h_{rk}, \forall O_{ij} \text{ s.t. } mr_{ij} = M_r \tag{4.31}$$

$$\beta_{ijk}[C_{ij} - t_{ij} + M(1 - Z_{ij,rk})] \geq \beta_{ijk}[p_{ij} + p_{rk}^{'} + \alpha_{ij}(S_{rk} - t_{ij})] \qquad \forall h_{rk}, \forall O_{ij} \text{ s.t. } mr_{ij} = M_r \tag{4.32}$$

$$\beta_{ijk}[-t_{ij} + S_{rk} - M(1 - Z_{ij,rk})] \leq \beta_{ijk}(p_{ij} - \varepsilon) \qquad \forall h_{rk}, \forall O_{ij} \text{ s.t. } mr_{ij} = M_r \tag{4.33}$$

$$Y_{ij,r(k+1)} \geq Y_{ij,rk} \qquad \forall h_{rk}, \forall O_{ij} \text{ s.t. } mr_{ij} = M_r \tag{4.34}$$

$$C_{max} \geq C_{in_i} \qquad i = 1,..,n \tag{4.35}$$

$$C_{ij} - t_{ij} \geq p_{ij} \qquad i = 1,..,n; j = 1,..,n_i \qquad (4.36)$$
$$t_{ij} \geq 0 \qquad i = 1,..,n; j = 1,..,n_i \qquad (4.37)$$
$$C_{ij} \geq 0 \qquad i = 1,..,n; j = 1,..,n_i \qquad (4.38)$$
$$X_{ij,i'j'} \in \{0,1\} \qquad \forall\ (O_{ij} \neq O_{i'j'}) \text{ s.t. } mr_{ij} = mr_{i'j'} \qquad (4.39)$$
$$Y_{ij,rk} \in \{0,1\} \qquad \forall h_{rk}, \forall O_{ij}\ s.t.\ mr_{ij} = M_r \qquad (4.40)$$
$$Z_{ij,rk} \in \{0,1\} \qquad \forall h_{rk}, \forall O_{ij}\ s.t.\ mr_{ij} = M_r \qquad (4.41)$$

where M is a very large value.

Here also the objective is to minimize the makespan C_{max} ([4.24]). As in Constraint [4.2], conjunctive Constraint [4.25] ensures the respect of job routings. The disjunctive constraints [4.26] and [4.27] (which replace Constraints [4.3] and [4.4]) guarantee the non-overlapping of operations O_{ij} and $O_{i'j'}$ that must be processed on the same machine. The disjunctive constraint [4.28] ensures that, if $Y_{ij,rk} = 0$, then operation O_{ij} must be processed after unavailability period h_{rk}. Otherwise, the constraint is always satisfied. It is similar to Constraint [4.5]. Constraint [4.29] expresses the fact that, when $Y_{ij,rk} = 1$, operation O_{ij} must start before the starting date of unavailability period h_{rk}. But, depending on the values of β_{ijk} and $Z_{ij,rk}$, it can either start and finish before h_{rk} or start before h_{rk} and finish after. When $\beta_{ijk} = 0$ (no preemption allowed), O_{ij} starts and finishes before the starting date of h_{rk} whatever the value of $Z_{ij,rk}$. But when $\beta_{ijk} = 1$ (preemption allowed), it depends on the value of $Z_{ij,rk}$. When $Z_{ij,rk} = 0$, operation O_{ij} must start and finish before h_{rk} whereas, when $Z_{ij,rk} = 1$, O_{ij} starts before h_{rk} and finish after (O_{ij} is interrupted by h_{rk}). Note that, when $Y_{ij,rk} = 0$, Constraint [4.29] is always satisfied. Constraint [4.30] expresses the possibility of interrupting O_{ij} when it starts before h_{rk} (see Figure (4.1)). Constraints [4.31] and [4.32] provide lower bounds for $C_{ij} - t_{ij}$ which corresponds to the duration of operation O_{ij} on machine M_r. Indeed, when O_{ij} is not interrupted by h_{rk}, this duration is at least equal (actually is equal) to the processing time p_{ij}. But when, O_{ij} is interrupted by h_{rk}, this duration is at least equal (actually is equal) to the processing time p_{ij} increased by the duration p'_{rk} of the unavailability period and the proportion of the operation to redo ($p_{ij} + p'_{rk} + \alpha_{ij}(S_{rk} - t_{ij})$). This is because:

$$C_{ij} = t_{ij} + (S_{rk} - t_{ij}) + p'_{rk} + (p_{ij} - (S_{rk} - t_{ij})) + \alpha_{ij}(S_{rk} - t_{ij})$$

where $(S_{rk} - t_{ij})$ represents the part of O_{ij} processed before the beginning of h_{rk}, $(p_{ij} - (S_{rk} - t_{ij}))$ the part remaining to carry out and $\alpha_{ij}(S_{rk} - t_{ij})$ the part to redo.
This happens only when $t_{ij} < S_{rk} < t_{ij} + p_{ij}$, which is replaced by $t_{ij} + \varepsilon \leq S_{rk} \leq t_{ij} + p_{ij} - \varepsilon$ (Constraints [4.29], [4.33]) and $\beta_{ijk} = 1$ (preemption is allowed). This is due to the fact that introducing strict inequalities in a model makes it non linear. Coefficient ε must be chosen as small as possible to guarantee the equivalence between the two expressions. In addition, as in Constraint [4.7], Constraint [4.34] expresses the transitive relationship between an operation and unavailability periods on the same machine. Constraint [4.35] (which replaces Constraint [4.8]) indicates that the schedule cannot be completed before the end of the last operation of each job. Constraint [4.36]) represents a bound for completion dates of operations. It is relevant when the machine is continuously available, otherwise it can be redundant. Constraints from

[4.37] to [4.41] provide bounds on the variables of the model.

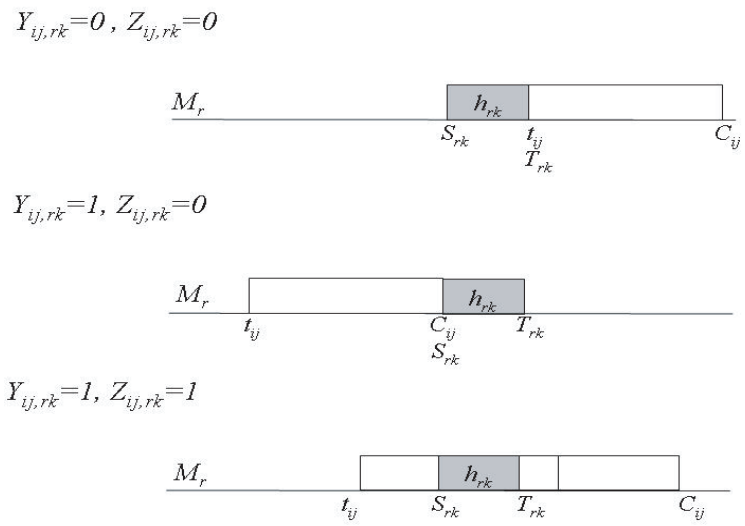

Figure 4.1: Position of operation O_{ij} depending on values of $Y_{ij,rk}$ and $Z_{ij,rk}$.

4.1.2.2 Flexibility on machine unavailability periods

We add flexibility on starting dates of the unavailability periods by adding Constraints [4.13] and [4.14] to the model. We also introduce flexibility on the duration of unavailability period h_{rk} by assuming that it can vary in a set of V_{rk} values $\{v_{rk1}, v_{rk2}, \cdot\cdot, v_{rkl}, \cdot\cdot, v_{rkV_{rk}}\}$. We denote p'_{rkl} the binary variable which defines if v_{rkl} is part of the duration of h_{rk}; i.e. p'_{rkl} is equal to 1 if v_{rkl} is chosen and 0 otherwise. We use binary variables to express durations of unavailability periods instead of integer variables because they are easier to express, and we believe the models with binary variables are easier to solve by standard solvers. The values of durations can be different from an unavailability period to another, which means that each unavailability period can have its own policy. We then add to the model the following constraints:

$$t_{ij} - S_{rk} + MY_{ij,rk} \geq \sum_{l=1}^{V_{rk}} v_{rkl} p'_{rkl} \qquad \forall h_{rk}, \forall O_{ij}$$
$$s.t.\ mr_{ij} = M_r \qquad (4.42)$$

$$\beta_{ijk}[C_{ij} - t_{ij} + M(1 - Z_{ij,rk})] \geq \beta_{ijk}[p_{ij} + \sum_{l=1}^{V_{rk}} v_{rkl} p'_{rkl} + \alpha_{ij}(S_{rk} - t_{ij})] \qquad \forall h_{rk}, \forall O_{ij}$$
$$s.t.\ mr_{ij} = M_r \qquad (4.43)$$

$$\sum_{k=1}^{m_r} \sum_{l=1}^{V_{rk}} p'_{rkl} \geq n_r \qquad r = 1, .., m \qquad (4.44)$$

$$\sum_{k=1}^{m_r} \sum_{l=1}^{V_{rk}} v_{rkl} p'_{rkl} \geq h_r \qquad r = 1, .., m \qquad (4.45)$$

$$p'_{rkl} \in \{0, 1\} \qquad r = 1, .., m;$$

$$k = 1, .., m_r; \quad l = 1, .., V_{rk} \tag{4.46}$$

Hence, in the above model, Constraint [4.28] is replaced by Constraint [4.42] and Constraint [4.32] by Constraint [4.43]. Constraint [4.44] expresses the fact that, on each machine M_r, the number of unavailability periods should not be smaller than n_r, and Constraint [4.45] expresses the fact that the total unavailability duration on each machine M_r should not be smaller than h_r. This implies that, even if the lower bound on the total unavailability duration is satisfied with less unavailability periods than n_r, it must be balanced on at least n_r unavailability periods.

Remarks:

1. It is possible to have more than one value v_{rkl} for an unavailability period.
   To have at most one value, the following constraint must be added to the model:
   $$\sum_{l=1}^{V_{rk}} p'_{rkl} \leq 1$$
   Each of the two policies needs an adjustment in the definition of the set of values v_{rkl} and the constraints.
2. It is possible that no value is associated to an unavailability period to set the priority to job production instead of machine unavailability.
3. In Constraint [4.44], if at least two unavailability periods are pasted, number n_r means number of parts of machine unavailability. To refer to n_r as the number of unavailability periods, it is necessary to add the constraint of remark 1.

4.1.2.3 Numerical results

The test results for the general model are summarized in Tables 4.8 through 4.11 depending on the considered case. Their structure is the same as the one of Table 4.1 except that Columns 2 to 5 represent test results for fixed unavailability periods, Columns 6 to 9 summarize test results for non-fixed starting dates and fixed durations of unavailability periods, and Columns 10 to 13 give test results for non-fixed stating dates and non-fixed durations of unavailability periods.

Comparing the results for fixed unavailability periods and non-fixed starting dates and fixed durations of unavailability periods, Table 4.8 summarizes the test results for the non-preemptive case. This case is obtained by setting all coefficients β_{ijk} to 0. For the other cases (resumable, non resumable and semi-resumable), all coefficients β_{ijk} are set to 1, whereas coefficients α_{ij} are respectively set to 0, 1 and 0.5. The associated results are summarized in Tables 4.9, 4.10 and 4.11. Note that we provide results for these cases because they are the most representative. We set the value of ε to 1. For testing the flexibility on durations of unavailability periods, we set $n_r = 1$ and h_r equal to half amount of unavailability of the machine M_r.

The standard solver (CPLEX 10) solved optimally benchmarks up to 10 machines and 10 jobs for the non-preemptive case and up to 5 machines and 5 jobs for the other cases in very short time. But in general it takes a long time to solve problems to optimality, and only a feasible solution is often provided.

The results confirm that it can be effective to allow the starting dates of unavailability periods to vary within some time windows. From Tables 4.8 and 4.10, it can be verified that, due to the penalty induced by reprocessing the entire operation when it is interrupted by an unavailability period in a deterministic context, it is more relevant not to interrupt the operation. Indeed, for the non-resumable case, the test results show that no interruption of an operation by an unavailability period is allowed except if the slot on the machine, before the unavailability period, cannot be used to process another operation. Hence, the non-preemptive and non-resumable cases are equivalent. Tables 4.8, 4.9 and 4.11 show that it is more interesting to allow preemption between operations and unavailability periods because less time is needed to complete all jobs. Indeed, for the resumable case, as interruption is allowed with no penalty, all operations can be processed as early as possible, and then less idle time is left on machines; for the semi-resumable case there is a gain in allowing preemption with a penalty smaller than 1 (which corresponds to 100 percent of the operation processing time).

Table 4.8: Test results of the general disjunctive model in non-preemptive case.

Problem Problem	Fixed unavailability periods				Non-fixed starting dates and fixed durations of unavailability periods				Non-fixed starting dates and non-fixed durations of unavailability periods			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	895	895	0	1.93	825	825	0	0.09	705	705	0	0.24
5m5n2	1096	1096	0	0.03	1076	1076	0	0.02	859	859	0	0.20
5m5n3	1070	1070	0	0.16	1034	1034	0	0.23	848	848	0	0.29
5m5n4	1147	1147	0	0.15	1108	1108	0	0.27	959	959	0	0.24
5m5n5	1202	1202	0	0.2	1182	1182	0	0.51	951	951	0	0.51
5m10n1	1361	1361	0	65.08	1300	1300	0	43.09	1058	1190	11.09	3595.96
5m10n2	1435	1455	1.37	3167.62	1271	1400	9.21	1519.4	1029	1290	20.23	3506.37
5m10n3	1527	1607	4.98	2822.06	1274	1587	19.72	1939.35	1077	1409	23.56	3456.91
5m10n4	1537	1537	0	609.85	1478	1492	0.94	1324.87	1171	1376	14.86	3545.37
5m10n5	1355	1415	4.24	3437.75	1372	1372	0	174.72	1172	1243	5.71	3597.20
10m10n1	1887	1948	3.13	2509.77	1771	1859	4.73	2073.96	1595.65	1700	6.14	3592.41
10m10n2	1917	1917	0	64.89	1839	1839	0	664.17	1542	1696	9.08	3582.08
10m10n3	1855	1875	1.07	3028.38	1773	1773	0	466.46	1573	1573	0	2588.84
10m10n4	1772	1772	0	136.23	1690	1690	0	669.87	1438	1585	9.27	3587.04
10m10n5	1847	1975	6.48	2858.81	1725	1833	5.89	1814.3	1450	1680	13.69	3545.34
10m15n1	1717	2089	17.81	1750.16	1591	2103	24.35	2708.88	1448	1929	24.94	3574.74
10m15n2	1880	2316	18.83	2562.63	1703	2388	28.69	1893.73	1429	2095	31.79	3524.85
10m15n3	1721	2261	23.88	1844.29	1637	2221	26.29	2645.88	1418	2045	30.66	3527.62
10m15n4	1744	2254	22.63	1799.69	1488	2120	29.81	2679.33	1362	1999.50	31.88	3554.80
10m15n5	1746	2282	23.47	1640.13	1627	2225	26.88	2807.98	1499	2066	27.44	3555.66

Table 4.9: Test results of the general disjunctive model in resumable case.

Problem	Fixed unavailability periods				Non-fixed starting dates and fixed durations of unavailability periods				Non-fixed starting dates and non-fixed durations of unavailability periods			
Problem	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	845	845	0	1.19	825	825	0	0.86	700	700	0	3.55
5m5n2	997	997	0	0.3	977	977	0	0.49	838	838	0	1.23
5m5n3	1020	1020	0	2.35	1020	1020	0	4.45	800	800	0	1.07
5m5n4	1135	1135	0	1.4	1108	1108	0	1.69	910	910	0	0.89
5m5n5	1108	1108	0	3.22	1108	1108	0	14.67	951	951	0	10.82
5m10n1	1021	1322	22.77	2804.52	920	1316	30.09	1919.40	890	1177	24.38	3496.37
5m10n2	1051	1400	24.93	2087.99	970	1400	30.71	2633.12	950	1290	26.36	3449.11
5m10n3	989.20	1583	37.51	2320.4	1039.62	1563	33.49	2350.26	960	1389	30.89	3489.86
5m10n4	1047	1492	29.83	2314.88	1090	1492	26.94	2309.88	972	1376	29.36	3498.93
5m10n5	1024	1386	26.12	2219.43	1084	1372	20.99	2594.16	930	1243	25.18	3451.59
10m10n1	1578	1907	17.25	2629.69	1498	1912	21.65	2052.89	1410	1700	17.06	3561.88
10m10n2	1561	1822	14.32	2607.49	1582	1915	17.39	1834.03	1459	1690	13.67	3535.8
10m10n3	1560	1815	14.05	2015.76	1458	1726	15.53	2463.87	1366.19	1580	13.53	3552.65
10m10n4	1455	1755	17.09	2145.85	1435	1701	15.64	2386.44	1320	1565	15.65	3552.26
10m10n5	1641	1870	12.25	2084.82	1443.83	1839	21.49	2404.60	1345.70	1721	21.81	3535.78
10m15n1	1563	2310	32.34	2193.86	1473	2087	29.42	2453.20	1340	3475	61.44	3573.83
10m15n2	1655	2460	32.72	2420.44	1527	2450	37.67	2198.51	1328.42	2150	38.21	3532.5
10m15n3	1415	2313	38.82	2491.91	1413.97	4202	66.35	2121.47	1350	2047	34.05	3555.99
10m15n4	1457	6958	79.06	1858.05	1344.08	4941	72.80	2790.20	1300	1951	33.37	3542.84
10m15n5	1537.37	6392	75.95	2186.95	1492	2292	34.90	2368.06	1336.09	2321	42.43	3568.95

Table 4.10: Test results of the general disjunctive model in non-resumable case.

Problem Problem	Fixed unavailability periods				Non-fixed starting dates and fixed durations of unavailability periods				Non-fixed starting dates and non-fixed durations of unavailability periods			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	895	895	0	0.43	825	825	0	1.44	705	705	0	0.81
5m5n2	1096	1096	0	0.19	1076	1076	0	0.02	859	859	0	0.46
5m5n3	1070	1070	0	1.84	1034	1034	0	3.46	848	848	0	0.48
5m5n4	1147	1147	0	0.43	1108	1108	0	1.86	959	959	0	10.43
5m5n5	1202	1202	0	0.85	1182	1182	0	0.51	951	951	0	0.96
5m10n1	1230	1361	9.63	2470.88	1063	1302	18.36	2259.55	930	1190	21.85	3535.65
5m10n2	1291	1455	11.27	2357.62	1121.30	1400	19.91	2303.00	923	1290	28.45	3490.59
5m10n3	1131	1637	30.91	2448.72	1009	1578	36.06	2238.09	894	1415	36.82	3526.62
5m10n4	1258	1537	18.15	2364.82	1228	1492	17.69	2347.14	1066	1376	22.53	3577.28
5m10n5	1134	1415	19.86	2434.56	992	1377	27.96	2276.81	980	1243	21.16	3538.07
10m10n1	1627	1987	18.12	2001.49	1592	1912	16.74	2423.92	1397.01	1764	20.80	3582.09
10m10n2	1736	1917	9.44	2236.83	1697	1874	9.45	2202.90	1470	1728	14.93	3580.69
10m10n3	1725	1875	8.00	2030.33	1525	1773	13.99	2385.17	1390	1573	11.63	3577.77
10m10n4	1688	1772	4.74	2148.22	1515	1690	10.36	2307.29	1320	1586	16.77	3571.60
10m10n5	1847	1933	4.45	2442.21	1621	1880	13.78	2052.89	1349	1694	20.37	3528.93
10m15n1	1567	2209	29.06	2192.23	1493.25	2137	30.12	2294.99	1290	1949	33.81	3573.47
10m15n2	1699	2326	26.96	1807.45	1679	2395	29.90	2761.13	1340.26	2168	38.18	3571.42
10m15n3	1661	2213	24.94	2639.09	1407.61	2403	41.42	1932.06	1282.42	2120	39.51	3581.37
10m15n4	1537	2212	30.52	2177.27	1406	2166	35.09	2377.87	1258.07	2112	40.43	3570.96
10m15n5	1639	2342	30.02	2577.29	1527	2146	28.84	1921.04	1398	2097	33.33	3573.51

When durations of unavailability periods are not fixed, we assumed that the minimum number of unavailability periods on each machine is 1; and the minimum amount of unavailability of each machine is equal to half the total unavailability duration. The test results show that it is relevant to introduce flexibility on the durations of unavailability periods, since better results are obtained for the makespan. However, it induces more complexity expressed by the CPU time, and less instances of Classes (5,10) and (10,10) are solved to optimality.

4.1.3 Number of constraints and variables of the models

Let us denote:

n_{ir} the number of operations of job J_i to process on machine M_r,
$N = \sum_{i=1}^{n} n_i$,
$N_h = \sum_{r=1}^{m} m_r$,
$N_X = \sum_{r=1}^{m} \sum_{i=1}^{n-1} \sum_{i'=i+1}^{n} n_{ir} n_{i'r}$,
$N_Y = \sum_{r=1}^{m} \sum_{i=1}^{n} n_{ir} m_r$.

4.1.3.1 Non-preemptive problem

Disjunctive formulation

Fixed starting dates for unavailability periods The number of variables in this model are: 1 variable C_{max}, N variables t_{ij}, N_X variables $X_{ij,i'j'}$ and N_Y variables $Y_{ij,rk}$. The number of constraints are: $N - n$ constraints [4.2], $2N_X$ constraints [4.3] and [4.4], $3N_Y$ constraints [4.5]-[4.7], n constraints [4.8] and 1 constraint [4.9].

Non-fixed starting dates for unavailability periods Comparing to the case of fixed starting dates, this model requires N_h additional variables.

Without unavailability periods Compared to the case of fixed starting dates, this model requires N_Y less variables (for $Y_{ij,rk}$) and $3N_Y$ less constraints (for Constraints [4.5]-[4.7]).

Time-indexed formulation The number of variables in this model are: 1 variable C_{max}, $\sum_{i=1}^{n} \sum_{j=1}^{n_i} (Lt_{ij} - Et_{ij} + 1)$ variables x_{ij}^t and $\sum_{r=1}^{m} \sum_{k=1}^{m_r} (LS_{rk} - ES_{rk} + 1)$ variables y_{rk}^t. The number of constraints in this model are: N constraints [4.16], N_h constraints [4.17], $N - n$ constraints [4.18] and mT constraints [4.20].

The comparison on the numbers of variables and constraints between the disjunctive and time-indexed formulations is made in the numerical experiments (see Section 4.2.1.3).

4.1.3.2 General disjunctive model

The number of variables in this model are: 1 variable C_{max}, $2N$ variables t_{ij} and C_{ij}, N_X variables $X_{ij,i'j'}$ and $2N_Y$ variables $Y_{ij,rk}$ and $Z_{ij,rk}$. The number of constraints are: $N - n$

Table 4.11: Test results of the general disjunctive model in semi-resumable case.

Problem Problem	Fixed unavailability periods				Non-fixed starting dates and fixed durations of unavailability periods				Non-fixed starting dates and non-fixed durations of unavailability periods			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	866.62	866.62	0	1.17	825.00	825.00	0	0.43	705	705	0	0.49
5m5n2	1042.25	1042.25	0	1.65	1000.00	1000.00	0	0.49	848.50	848.50	0	0.82
5m5n3	1046.50	1046.50	0	3.56	1031.50	1031.50	0	3.51	831	831	0	1.49
5m5n4	1137.00	1137.00	0	1.62	1108.00	1108.00	0	3.72	934.50	934.50	0	0.56
5m5n5	1159.50	1159.50	0	2.51	1126.50	1145.00	0	4.4	951	951	0	2.82
5m10n1	1081.00	1378.50	21.58	2452.69	1017.00	1339.25	24.06	2273.56	930	1181.50	21.99	3502.93
5m10n2	1148.00	1452.75	20.98	2388.3	1001.00	1400.00	28.50	2230.54	891	1290	30.93	3482.84
5m10n3	1092.00	1625.50	32.82	2504.31	1003.00	1578.00	36.43	2169.74	927	1409	34.21	3509.09
5m10n4	1348.00	1503.50	10.34	2615.14	1160.00	1496.50	22.49	2025.93	980	1376	28.78	3506.57
5m10n5	1012.64	1406.00	27.98	2453.44	1014.00	1372.00	26.09	2296.19	895.55	1243	27.95	3482.83
10m10n1	1631.00	1906.00	14.43	3058.65	1522.00	1920.00	20.73	1503.73	1377	1712	19.57	3539.04
10m10n2	1654.75	1957.50	15.47	1986.22	1632.00	1864.62	12.48	2454.36	1399.41	1712.50	18.28	3508.76
10m10n3	1642.00	1825.25	10.04	2078.23	1535.22	1809.50	15.16	2315.15	1380	1608	14.18	3535.94
10m10n4	1584.00	1780.00	11.01	1818.02	1435.00	1714.00	16.28	2625.75	1320	1589	16.93	3568.31
10m10n5	1645.00	1938.37	15.14	1966.90	1475.00	1871.00	21.17	2446.70	1320	1677	21.31	3488.19
10m15n1	1563.00	2107.00	25.82	2717.59	1487.00	2199.00	32.38	1923.29	1300	1974	34.14	3571.41
10m15n2	1655.00	2366.00	30.05	2508.09	1548.00	2413.00	35.83	2049.56	1320	2129.50	38.01	3567.09
10m15n3	1459.31	2271.00	35.74	1758.44	1489.50	2269.00	34.35	2846.19	1306.18	2154.50	39.37	3580.40
10m15n4	1468.50	2214.00	33.67	2090.79	1441.00	2562.00	43.75	2512.73	1254.42	2112	40.61	3580.05
10m15n5	1629.00	2272.25	28.31	2567.89	1546.00	2183.00	29.20	2020.38	1326.52	2060	35.61	3574.87

constraints [4.25], $2N_X$ constraints [4.26] and [4.27], $7N_Y$ constraints [4.28]-[4.34], n constraints [4.35] and N constraints [4.36].

Hence, the generalization of the basic model increases the number of variables by $N + N_Y$ and the number of constraints by $N + 4N_Y$.

Introducing flexibility on durations of unavailability periods leads to $\sum_{r=1}^{m} \sum_{k=1}^{m_r} v_{rk}$ additional variables p'_{rkl}. The number of additional constraints [4.42], [4.43], [4.44], [4.45] and [4.46] are respectively N_y, N_y, m, m and $\sum_{r=1}^{m} \sum_{k=1}^{m_r} v_{rk}$.

4.2 Model extensions

4.2.1 Flexible job shop problem with resource availability constraints

In this section, we deal with the flexible job shop problem when resources are not continuously available. It is an extension of the job shop problem under resource availability constraints. Its specificity is that a job operation can be processed by more than one machine but needs only one. This offers more flexibility to the production system but induces more complexity. Indeed, in addition to the sequencing problem, an assignment problem of the operations to the machines occurs. The mathematical model we propose is a generalization of the general disjunctive model presented in Section 4.1.2.

4.2.1.1 Problem definition

The flexible job shop scheduling problem with resource availability constraints can be defined as a set of n jobs $J = \{J_1, J_2, \cdot\cdot, J_n\}$ to be processed on a set of m machines $M = \{M_1, M_2, \cdot\cdot, M_m\}$. Each job J_i is composed of a linear sequence of n_i operations $\{O_{i1}, O_{i2}, \cdot\cdot, O_{ij}, \cdot\cdot, O_{in_i}\}$. Each machine can process only one operation at a time and each operation O_{ij} needs only one machine M_a in a set A_{ij} of machines during p^a_{ij} time units (the processing time of an operation depends on the machine on which it is processed). Note that, for the classical job shop scheduling problem, the machine is defined a priori.

There are m_r unavailability periods $\{h_{r1}, h_{r2}, \cdot\cdot, h_{rk}, \cdot\cdot, h_{rm_r}\}$ on each machine M_r. The starting date S_{rk} of unavailability period h_{rk} of duration p'_{rk} is known in advance and can vary in the interval $[ES_{rk}, LS_{rk}]$. The machine on which operation O_{ij} is processed is denoted mr_{ij}.

The objective is to assign a machine to operation O_{ij}, and to determine its starting date t_{ij} and completion date C_{ij}. The objective function is to minimize the makespan C_{max}.

The flexible job shop problem is NP-hard when the number of machine is two or more and the number of jobs is three or more $J(MPM)2|n = 3|C_{max}$.

4.2.1.2 Disjunctive model

This formulation is deduced from the one we presented in Section 4.1.2 for the job shop scheduling problem with resource availability constraints and the one proposed in Roux [Rou97] for

complex shop scheduling problems. In this latter an operation may need more than one resource to be processed. Each resource must be chosen in a set of values. The job routings are non-linear which means that an operation can have more than one predecessor and one successor. The resources are assumed to be continuously available.

We introduce to the model the following additional variables:

x_{ij}^a: A binary variable which is equal to 1 if operation O_{ij} is processed on machine M_a and 0 otherwise,

p_{ij}: Processing time of operation O_{ij} which depends on the selected machine,

$W_{ij,rk}$: A binary variable which represent the linearization of $Z_{ij,rk} \times p_{ij}$. Hence, if $Z_{ij,rk} = 0$ (resp. $Z_{ij,rk} = 1$), then $W_{ij,rk} = 0$ (resp. $W_{ij,rk} = p_{ij}$).

The model is as follows:

$$\min C_{max} \tag{4.47}$$

$$\sum_{a \in A_{ij}} x_{ij}^a = 1 \qquad i = 1,..,n; j = 1,..,n_i \tag{4.48}$$

$$p_{ij} \geq \sum_{a \in A_{ij}} p_{ij}^a x_{ij}^a \qquad i = 1,..,n; j = 1,..,n_i \tag{4.49}$$

$$t_{i(j+1)} \geq C_{ij} \qquad i = 1,..,n; \; j = 1,..,n_i - 1 \tag{4.50}$$

$$t_{ij} + MX_{ij,i'j'} \geq C_{i'j'} - M(2 - x_{ij}^a - x_{i'j'}^a) \qquad \forall a \in (A_{ij} \cap A_{i'j'}), \; \forall (O_{ij} \neq O_{i'j'}) \tag{4.51}$$

$$t_{i'j'} + M(1 - X_{ij,i'j'}) \geq C_{ij} - M(2 - x_{ij}^a - x_{i'j'}^a) \qquad \forall a \in (A_{ij} \cap A_{i'j'}), \; \forall (O_{ij} \neq O_{i'j'}) \tag{4.52}$$

$$t_{ij} - S_{rk} + MY_{ij,rk} \geq p_{rk}^{'} - M(1 - x_{ij}^a) \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.53}$$

$$-t_{ij} + S_{rk} + M(1 - Y_{ij,rk}) \geq p_{ij} - \beta_{ijk} W_{ij,rk} + \varepsilon \beta_{ijk} Z_{ij,rk} - M(1 - x_{ij}^a) \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.54}$$

$$\beta_{ijk} Z_{ij,rk} \leq \beta_{ijk}[Y_{ij,rk} + M(1 - x_{ij}^a)] \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.55}$$

$$\beta_{ijk} W_{ij,rk} \leq \beta_{ijk}[MZ_{ij,rk} + M(1 - x_{ij}^a)] \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.56}$$

$$\beta_{ijk} W_{ij,rk} \leq \beta_{ijk}[p_{ij} + M(1 - x_{ij}^a)] \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.57}$$

$$\beta_{ijk} W_{ij,rk} \geq \beta_{ijk}[p_{ij} - M(1 - Z_{ij,rk}) - M(1 - x_{ij}^a)] \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.58}$$

$$C_{ij} - t_{ij} + \beta_{ijk} MZ_{ij,rk} \geq p_{ij} - M(1 - x_{ij}^a) \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.59}$$

$$\beta_{ijk}[C_{ij} - t_{ij} + M(1 - Z_{ij,rk})] \geq \beta_{ijk}[p_{ij} + p_{rk}^{'} + \alpha_{ij}(S_{rk} - t_{ij}) - M(1 - x_{ij}^a)] \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.60}$$

$$\beta_{ijk}[-t_{ij} + S_{rk} - M(1 - Z_{ij,rk})] \leq \beta_{ijk}[p_{ij} - \varepsilon + M(1 - x_{ij}^a)] \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.61}$$

$$Y_{ij,r(k+1)} \geq Y_{ij,rk} - M(1 - x_{ij}^a) \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \tag{4.62}$$

$$C_{max} \geq C_{in_i} \qquad i = 1,..,n \tag{4.63}$$

$$C_{ij} \geq t_{ij} + p_{ij} \qquad i = 1,..,n; j = 1,..,n_i \tag{4.64}$$

$$t_{ij} \geq 0 \qquad i = 1,..,n; j = 1,..,n_i \tag{4.65}$$

$$C_{ij} \geq 0 \qquad i = 1, .., n; j = 1, .., n_i \quad (4.66)$$

$$p_{ij} \geq 0 \qquad i = 1, .., n; j = 1, .., n_i \quad (4.67)$$

$$S_{rk} \geq ES_{rk} \qquad r = 1, .., m; \; k = 1, .., m_r \quad (4.68)$$

$$S_{rk} \leq LS_{rk} \qquad r = 1, .., m; \; k = 1, .., m_r \quad (4.69)$$

$$X_{ij,i'j'} \in \{0, 1\} \qquad \forall a \in (A_{ij} \cap A_{i'j'}), \; \forall (O_{ij} \neq O_{i'j'}) \quad (4.70)$$

$$Y_{ij,rk} \in \{0, 1\} \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \quad (4.71)$$

$$Z_{ij,rk} \in \{0, 1\} \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \quad (4.72)$$

$$W_{ij,rk} \in \{0, 1\} \qquad \forall a \in A_{ij}, \forall h_{rk}, \forall O_{ij} \; s.t. \; a = M_r \quad (4.73)$$

$$x^a_{ij} \in \{0, 1\} \qquad i = 1, .., n; j = 1, .., n_i, \; \forall a \in A_{ij} \quad (4.74)$$

The objective is makespan minimization ([4.47]). Constraint [4.48] expresses the fact that only one machine is chosen from A_{ij} to process operation O_{ij}. Constraint [4.49] determines the value of the processing time of operation O_{ij} depending on the selected machine. Conjunctive Constraint [4.50] ensures the respect of job routings (it is similar to Constraint [4.25]). Disjunctive Constraints [4.51] and [4.52] guarantee the non overlapping of operations O_{ij} and $O_{i'j'}$ that must be processed on the same machine, i.e. that operation O_{ij} is either processed before or after operation $O_{i'j'}$. If at least one of operations O_{ij} and $O_{i'j'}$ is not processed on machine M_a ($x^a_{ij} = 0$ or $x^a_{i'j'} = 0$) the constraints are always satisfied. If the two operations are processed on M_a ($x^a_{ij} = 1$ and $x^a_{i'j'} = 1$), O_{ij} is processed after $O_{i'j'}$ if $X_{ij,i'j'} = 0$ and O_{ij} is processed before $O_{i'j'}$ if $X_{ij,i'j'} = 1$ (they replace Constraints [4.26] and [4.27]). Similarly, Constraints [4.53] and [4.54] ensure that operation O_{ij} is either processed before or after unavailability period h_{rk} when O_{ij} is processed on machine M_a ($x^a_{i'j'} = 1$); otherwise $x^a_{i'j'} = 0$ and then the constraints are always satisfied. Disjunctive constraints are associated to resource constraints which express the fact that, when a machine is available, it can process only one operation at a time (they replace Constraints [4.28] and [4.29]). Constraints [4.55]) through [4.58]) replace Constraint [4.30]). They define the values of $Z_{ij,rk}$ and $W_{ij,rk}$. Constraints [4.59] and [4.60] provide the value of the completion date of operation O_{ij}. Constraints [4.54] and [4.61] (which replace Constraints [4.29] and [4.33]) express inequalities $t_{ij} + \varepsilon \leq S_{rk} \leq t_{ij} + p_{ij} - \varepsilon$. Constraint [4.62] implies that, if operation O_{ij} is processed on machine M_a ($x^a_{i'j'} = 1$), then it precedes unavailability period h_{rk} and then it precedes all unavailability periods which follow h_{rk}. Moreover, if it follows unavailability period $h_{r(k+1)}$, then it follows all unavailability periods which precede $h_{r(k+1)}$. Constraint [4.63] indicates that the schedule cannot be completed before the end of the last operation of each job. Constraint [4.64] provide a bound on completion date of operation (it is similar to constraint [4.36]).

Gao *et al.* [GSS06] propose a formulation for the non-preemptive flexible job shop scheduling problem with non-fixed availability constraints. It is quite similar to the precedent formulation

except for disjunctive constraints. Indeed the disjunctive constraints [4.26] and [4.27] associated to operations O_{ij} and $O_{i'j'}$ are multiplied by $x^a_{ij} \times x^a_{i'j'}$ which make them valid only when $x^a_{ij} = 1$ and $x^a_{i'j'} = 1$ (operations O_{ij} and $O_{i'j'}$ to be processed on machine M_a). And disjunctive constraints [4.28] through [4.34] associated to operation O_{ij} and unavailability period h_{rk} are multiplied by x^a_{ij} which make them valid only when $x^a_{ij} = 1$ (operation O_{ij} to be processed on machine M_a on which unavailability period h_{rk} is planned). Hence the model is non-linear since these constraints induce multiplication of variables which is a problem if we intend to solve the model by integer linear programming solvers.

4.2.2 Optimization criteria

The extensions of the models concern the sum of completion dates of jobs $\sum_{i=1}^{n} C_i$ and maximum lateness L_{max}. There is no hierarchy between the makespan C_{max} and $\sum_{i=1}^{n} C_i$. However, C_{max} can be made equivalent to L_{max}.

4.2.2.1 Minimization of the sum of completion dates of jobs $\sum_{i=1}^{n} C_i$

In the general disjunctive model, the objective function [4.24] is replaced by

$$min \sum_{i=1}^{n} C_{in_i}$$

and Constraint [4.35] is removed.

The minimization of the weighted sum of completion dates of jobs $\sum_{i=1}^{n} w_i C_i$ can also be used.

Tests results

The test results for initial benchmarks are presented in Tables 4.12 through 4.15. Tables 4.12 summarizes the test results for the non-preemptive problem. Columns 2 and 3 present represent respectively the lower bound and the CPU time of the linear relaxation. Columns 4 to 7 present the results for fixed starting dates of unavailability periods, whereas Columns 8 to 11 present the results for non-fixed starting dates of unavailability periods. Columns 4 and 8 correspond to the best lower bound, and Columns 5 and 9 to the objective function of the best solution. Columns 6 and 10 show the gap. Columns 7 and 11 give the CPU time (in seconds).

The test results for resumable, non-resumable and semi-resumable problems are respectively presented in Tables 4.14, 4.13 and 4.15. Columns 2 to 5 present the results for fixed starting dates of unavailability periods, whereas Columns 6 to 9 present the results non-fixed starting dates of unavailability periods. Columns 2 and 6 correspond to the best lower bound, and Columns 3 and 7 to the objective function of the best solution. Columns 4 and 8 show the gap. Columns 5 and 9 give the CPU time (in seconds).

Table 4.12: Test results for for non-preemptive disjunctive formulation for $\sum C_i$ minimization.

Problem	Linear relaxation		Fixed unavailability periods				Non-fixed unavailability periods			
	Bound	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	2190	0	3652	3652	0	0.04	3552	3552	0	0.06
5m5n2	2650	0	4669	4669	0	0.01	4412	4412	0	0.01
5m5n3	2520	0	4008	4008	0	0.04	3968	3968	0	0.12
5m5n4	2600	0	4584	4584	0	0.06	4242	4242	0	0.06
5m5n5	2530	0	4663	4663	0	0.07	4565	4565	0	0.26
5m10n1	4760	0	9850	9850	0	672.04	9535	9535	0	2563.70
5m10n2	4990	0.01	10637	10637	0	167.89	10256	10256	0	1913.03
5m10n3	5210	0.01	10809.40	11072	2.37	3600	9316.36	10997	15.28	3600
5m10n4	5270	0.01	10630	10630	0	565	10491	10491	0	679.57
5m10n5	4960	0.01	9706	9706	0	1566.81	9101	9101	0	194.75
10m10n1	10510	0.01	15629.28	16878	7.40	3600	15081.50	16085	6.24	3600
10m10n2	10370	0.01	15777	15777	0	547.82	15200	15200	0	2151.56
10m10n3	10050	0.01	15352	15352	0	646.60	14469.06	15164	4.58	3600
10m10n4	9540	0.01	14715	15374	4.29	3600	13401.79	15155	11.57	3600
10m10n5	10010	0.01	15625	15625	0	1237.39	14448.85	15136	4.54	3600
10m15n1	14880	0.03	20750.43	27199	23.71	3600	19967.56	27221	26.65	3600
10m15n2	15440	0.03	22652.38	29186	22.39	3600	20766.35	27977	25.77	3600
10m15n3	14740	0.03	u				u			
10m15n4	14990	0.03	20860.50	27423	23.93	3600	19066.68	27143	29.75	3600
10m15n5	15530	0.03	21465.87	28609	24.97	3600	20484.81	28312	27.65	3600

u: unknown

The observations made for the C_{max} minimization are still valid for $\sum_i C_i$. Indeed, the standard solver (CPLEX 10) solves optimally instances up to Class (10,10) for non-preemptive problems and for Class (5,5) for the other cases in very short time. For the other instances, only a feasible solution is often provided.

It is also observed that it can be effective to allow the starting dates of unavailability periods to vary within some time windows. Moreover, due to the penalty induced by reprocessing the entire operation when it is interrupted by an unavailability period in a deterministic context, it is more relevant not to interrupt the operation; and the non-preemptive and non-resumable cases are equivalent. It is also more interesting to allow preemption between operations and unavailability periods because less time is needed to complete all jobs.

Columns 2 of Table 4.12 show that the gap between the bound of the linear relaxation and the integer value of the solution of an instance is relatively high. These values are more useful in Chapter 6 to better appreciate the quality of the linear relaxation of the column generation approach. Comparing the tests results of the two criteria C_{max} and $\sum_i C_i$, the linear relaxation of $\sum_i C_i$ minimization problem is better than C_{max} because the solution gaps of the instances are smaller.

Table 4.13: Test results for for resumable disjunctive formulation for $\sum C_i$ minimization.

Problem	Fixed unavailability periods				Non-fixed unavailability periods			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	3577	3577	0	0.32	3356	3356	0	0.90
5m5n2	4385	4385	0	0.12	4174	4174	0	0.12
5m5n3	3912	3912	0	0.39	3892	3892	0	1.98
5m5n4	4136	4136	0	0.23	4136	4136	0	1.04
5m5n5	4443	4443	0	1.23	4417	4417	0	2.37
5m10n1	8187.32	9610	14.80	3600	7587.13	9484	20.00	3600
5m10n2	9211.53	10402	11.44	3600	8541.57	10364	17.58	3600
5m10n3	9423.79	11130	15.33	3600	8412.82	10710	21.45	3600
5m10n4	8768.07	10526	16.70	3600	8509.98	10439	18.48	3600
5m10n5	8509.46	9407	9.54	3600	8050.77	9101	11.54	3600
10m10n1	14122.53	16335	13.54	3600	13556.42	16397	17.32	3600
10m10n2	13753.73	15353	10.42	3600	13369.14	15190	11.99	3600
10m10n3	13872.90	15349	9.62	3600	13368.79	15261	12.40	3600
10m10n4	12727.49	15423	17.48	3600	12312.27	14876	17.23	3600
10m10n5	13601.78	15669	13.19	3600	12861.07	15273	15.79	3600
10m15n1	19289.33	27010	28.58	3600	18512.37	26762	30.83	3600
10m15n2	19742.42	29251	32.51	3600	19030.30	28685	33.66	3600
10m15n3	18537.02	27196	31.84	3600	17812.63	27221	34.56	3600
10m15n4	18553.83	27530	32.61	3600	18075.02	27264	33.70	3600
10m15n5	19514.55	27601	29.30	3600	19128.66	28000	31.68	3600

Table 4.14: Test results for non-resumable disjunctive formulation for $\sum C_i$ minimization.

Problem	Fixed unavailability periods				Non-fixed unavailability periods			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	3652	3652	0	0.17	3552	3552	0	0.31
5m5n2	4669	4669	0	0.09	4412	4412	0	0.09
5m5n3	4008	4008	0	0.25	3968	3968	0	1.21
5m5n4	4584	4584	0	0.12	4242	4242	0	0.32
5m5n5	4663	4663	0	1.09	4565	4565	0	1.70
5m10n1	8852.72	9850	10.12	3600	8266.28	9585	13.76	3600
5m10n2	9844.48	10643	7.50	3600	9669.78	10274	5.88	3600
5m10n3	9631.99	11072	13.01	3600	8912.11	10558	15.59	3600
5m10n4	9474.00	10630	10.87	3600	9043	10493	13.82	3600
5m10n5	9241.11	9706	4.79	3600	8434	9292	9.23	3600
10m10n1	14651	17117	14.41	3600	13850.57	16297	15.01	3600
10m10n2	14967.05	15777	5.13	3600	13722.79	15213	9.80	3600
10m10n3	14438.67	15644	7.70	3600	13816.39	15278	9.57	3600
10m10n4	13855.92	15374	9.87	3600	12718.02	15045	15.47	3600
10m10n5	14780.55	15719	5.97	3600	13594.08	15347	11.42	3600
10m15n1	20136.59	26886	25.10	3600	u			
10m15n2	u				19682.12	28004	29.72	3600
10m15n3	19694.17	28308	30.43	3600	18430.75	27669	33.39	3600
10m15n4	19551.76	27763	29.58	3600	18226.26	28015	34.94	3600
10m15n5	u				19226.97	27763	30.75	3600

u: unknown

Table 4.15: Test results for semi-resumable disjunctive formulation for $\sum C_i$ minimization.

Problem	Fixed unavailability periods				Non-fixed unavailability periods			
	Best bound	Best solution	Gap (%)	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	3613	3613	0	0.42	3366.50	3366.50	0	0.34
5m5n2	4601	4601	0	0.15	4218.25	4218.25	0	0.14
5m5n3	3938	3938	0	1.07	3918	3918	0	1.54
5m5n4	4307	4307	0	0.23	4220.50	4220.50	0	1.15
5m5n5	4597.5	4597.5	0	1.14	4528	4528	0	2.07
5m10n1	8414.95	9843	14.51	3600	7977.12	9535	16.34	3600
5m10n2	9432.00	10440.5	9.66	3600	8928.63	10274	13.09	3600
5m10n3	9377.06	11106	15.57	3600	8267.92	10909	24.21	3600
5m10n4	9176.60	10588	13.33	3600	8711	10493	16.98	3600
5m10n5	8782.68	9515	7.70	3600	8334.02	9101	8.43	3600
10m10n1	14487.44	16806	13.80	3600	13763.20	16425	16.21	3600
10m10n2	14432.31	15572.7	7.32	3600	13337.52	15213	12.33	3600
10m10n3	13894.75	15352	9.49	3600	13393.43	15266	12.27	3600
10m10n4	13573	15411.2	11.93	3600	12445	15166.5	17.94	3600
10m10n5	14127.47	15595	9.41	3600	12951.39	15219	14.90	3600
10m15n1	19740.33	26993.2	26.87	3600	18793.01	28108	33.14	3600
10m15n2	u				19140.84	28712.5	33.94	3600
10m15n3	u				17763.26	26776.5	33.66	3600
10m15n4	18965.80	27277.2	30.47	3600	18276.79	26108	30	3600
10m15n5	19852.79	28805.2	31.08	3600	19042.34	27804	31.51	3600

u: unknown

4.2.2.2 Minimization of maximum lateness L_{max}

The maximum lateness L_{max} measures the worst violation of the due dates d_i.

In the general disjunctive model, the objective function [4.24] is replaced by

$$minL_{max},$$

and Constraint [4.35] is replaced by

$$L_{max} \geq C_{in_i} - d_i.$$

4.2.3 Constraints on job operations

Introduction of release dates on jobs r_i

We assume that the release date r_i is introduced for job J_i. It represents the earliest date at which job j can starts its processing. In the general disjunctive model, the following constraint is added: $t_{i1} \geq r_i$.

Introduction of deadlines on jobs $\tilde{d}_i$

The deadline $\tilde{d}_i$ is introduced for job J_i. It represents the date at which the processing of job J_i must be completed. In the general disjunctive model, the following constraint is added: $C_{in_i} \leq \tilde{d}_i$.

4.3 Conclusion

In this chapter, mathematical models were presented for the job shop scheduling problem with resource availability constraints. Two integer linear programming models were first presented when preemption is not allowed. Numerical experiments performed with a standard solver (CPLEX 10) show that the disjunctive formulation provides better results than the time-indexed formulation. We also introduced flexibility on the starting dates of unavailability periods.

A general disjunctive formulation was then proposed which models preemption between operations and unavailability periods, by integrating all possible cases of interruption of an operation by an unavailability period. In this model, we also added flexibility on the durations of unavailability periods. The numerical experiments showed that introducing flexibility is relevant; it is also relevant to allow preemption. But they also showed that a standard solver quickly reaches its limits for the resolution of these benchmarks. This is due to the strong complexity of the studied problems.

Mathematical modeling allowed for a better understanding of the problems through their mathematical expressions. The models showed how to deal with resource unavailability constraints; and they proved the relevance of introducing flexibility to the problems, not only to represent the reality in industry, but also to find better solutions to the problems. The models also allow for a better evaluation of the quality of solutions provided by approximation approaches, as it is difficult to provide theoretically good bounds for problems with machine unavailability periods.

The general disjunctive model was extended to consider other optimization criteria, to include other constraints on tasks and to model the flexible job shop scheduling problem with resource availability constraints.

Chapter 5

Approximation approaches

This chapter is dedicated to approximation methods (heuristics) as they aim to find good solutions for NP-hard combinatorial optimization problems in short.

We tackle the job shop scheduling problem with resource availability constraints. The originality of this approach is that it introduces flexibility on starting dates of unavailability periods and preemption of operations while constructing the schedule.

The structure of the chapter is the following: Section 5.1 present the construction methods we developed to tackle the job shop problem with flexible unavailability periods of resources and eventually preemptive operations. Section 5.2 discuss the way these construction heuristics, that are building blocks, are used in improving methods to obtain better results for the studied problem.

5.1 Construction methods

In this section, we present heuristics that construct a schedule based on various decision strategies. The choice of these strategies are related to how job operations and/or machines are prioritized, and how conflicts between jobs operations and machine unavailability periods are managed.

5.1.1 Elements of construction methods

5.1.1.1 General structure

The construction of schedules is based on the following elements:

1. Each time an operation has to be inserted, the heuristics try to schedule it as early as possible at the first availability period on the machine, starting from time zero if it is the first operation of the associated job, or based on the completion date of the immediately

previous operation in the job routing. The operation should not overlap an operation already scheduled or an unavailability period. If all the conditions are met, an operation can then be scheduled on a machine strictly before an operation already scheduled.

2. The flexibility on the starting dates of the unavailability periods: An unavailability period can move in its time window defined by its earliest and latest starting dates to create a sufficient idle time before or after the unavailability period to insert the operation as early as possible.

3. The preemption between a preemptive operation and a crossable period. The operation can be interrupted by the unavailability period if it cannot be completely processed before. This can possibly induce a penalty when the related coefficient is larger than or equal to 0.

We assume that each machine M_r has m_r unavailability periods. Hence, a schedule length is decomposed on machine M_r in $m_r + 1$ *availability periods* I_{r1}, I_{r2},.., $I_{r(m_r+1)}$. We prefer to use the term *interval* instead of availability period to avoid the confusion between an availability period and an unavailability period. The starting and end dates of the interval I_{rl} are respectively SI_{rl} and TI_{rl}.

5.1.1.2 Interval selection

To construct a schedule, each operation must be inserted in one of the intervals that define the machine availability. These intervals are not static, they change when scheduling is being built. Some intervals are modified and others are created or deleted. The most suitable situation is to reduce the interval lengths as much as possible because they correspond to idle times on the machines.

The insertion interval I_{rl} of the current operation O_{ij} is defined according to

- The completion date of the immediate previous operation in the job routing $O_{i(j-1)}$,
- The earliest availability interval of the machine,
- The length of the interval LI_{rl},
- The existence of an unavailability period h_{rk}, before the current interval I_{rl} (the starting date of the unavailability period corresponds to the end date of the precedent interval $S_{rk} = TI_{r(l-1)}$; the end date of the unavailability period corresponds to the starting date of the current interval $T_{rk} = SI_{rl}$); or between this I_{rl} and its immediate successor $I_{r(l+1)}$ (the starting date of the unavailability period corresponds to the end date of the current interval $S_{rk} = TI_{rl}$; and the end of the unavailability period corresponds to the starting date of the interval succeeding immediately to the interval $T_{rk} = SI_{r(l+1)}$), this can modify the length of the interval.

After each insertion, the intervals are updated. Note that an additional interval is created when the completion date of the operation preceding immediately the current operation in the job routing is larger than the starting date of the interval and the current operation finishes before the end of the current interval. Moreover, an interval can be deleted by moving an unavailability period when an operation is entirely inserted in all the interval.

5.1.1.3 Position of an operation relative to an unavailability period

When an operation is in presence of an unavailability period, two questions arise:

- Is it possible to move the unavailability period to insert the operation in the current interval and/or to process it earlier?

- Is it possible to interrupt the operation by the unavailability period to start the operation earlier?

Hence, in this section, we answer these questions by discussing cases of flexibility on the starting dates of the unavailability periods and for operation preemption relative to an unavailability period. Indeed, three cases are presented: A, B, and C. For each case, the conditions and how the intervals are updated are given.

Unlike the general mathematical model presented in Chapter 4, we consider here either the flexibility or the preemption but not both at the same time. This implies that there is a set of solution configurations that cannot be explored. But since the problem is NP-hard, it is also hard to find a good solution in reasonable time, in particular when preemption is allowed and flexibility on unavailability periods is introduced.

Note that cases A and B model flexibility; whereas case C models preemption (See Figures 5.1 and 5.2). The difference between cases A and B is that, in case A, we are interested in moving the unavailability periods placed at the beginning of the current interval and, in case B, at the end of the interval. Cases B and C are very close. The only difference is that in case C, the unavailability period cannot move and, as preemption is allowed, the operation is interrupted by the unavailability period and inserted in the current and following intervals.

Case A: Flexibility In this case, unavailability period h_{rk} precedes immediately interval I_{rl}, and can move. The following conditions must be satisfied:

- The previous operation in the job routing $O_{i(j-1)}$ must finish at most at the beginning of the current interval I_{rl} ($C_{i(j-1)} \leq SI_{rl}$)

- The starting date of the current interval I_{rl} corresponds to the end date of unavailability period h_{rk} ($SI_{rl} = T_{rk}$)

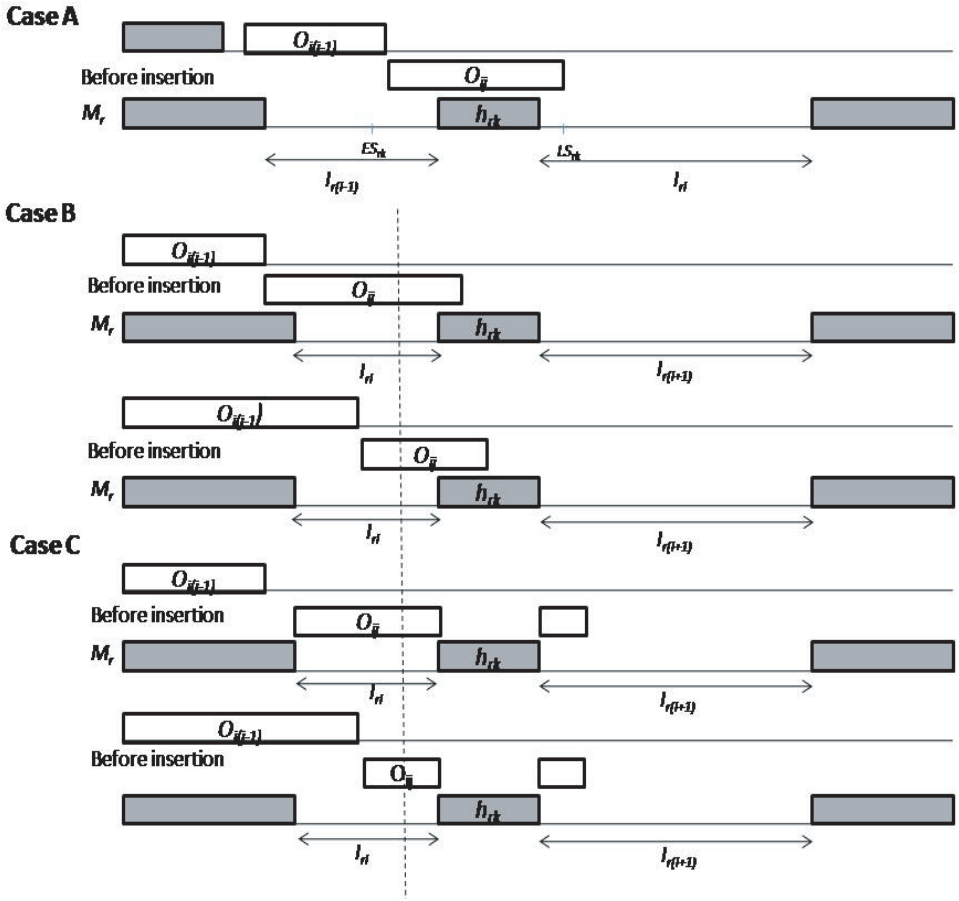

Figure 5.1: Comparison between Cases A, B and C before inserting operation O_{ij}

- The end date of the precedent interval $I_{r(l-1)}$ is the starting date of unavailability period h_{rk} $(TI_{r(l-1)} = S_{rk})$

As illustrated in Figure 5.3, two configurations are possible:

Case A.1

- Unavailability period h_{rk} can move to the left in its time window $(ES_{rk} < S_{rk})$
- The previous operation in the job routing $O_{i(j-1)}$ must finish before the beginning of the current interval I_{rl} $(C_{i(j-1)} < SI_{rl})$. Otherwise, even if unavailability period h_{rk} starts earlier in its time window, operation O_{ij} will start at SI_{rl}.

It is not necessary to move unavailability period h_{rk} to finish before the completion date of the previous operation in the job routing $O_{i(j-1)}$, because the current operation O_{ij} cannot be processed before $(C_{i(j-1)} \leq \tilde{T_{rk}} = \tilde{S_{rk}} + p'_{rk})$. h_{rk} cannot start before its earliest starting

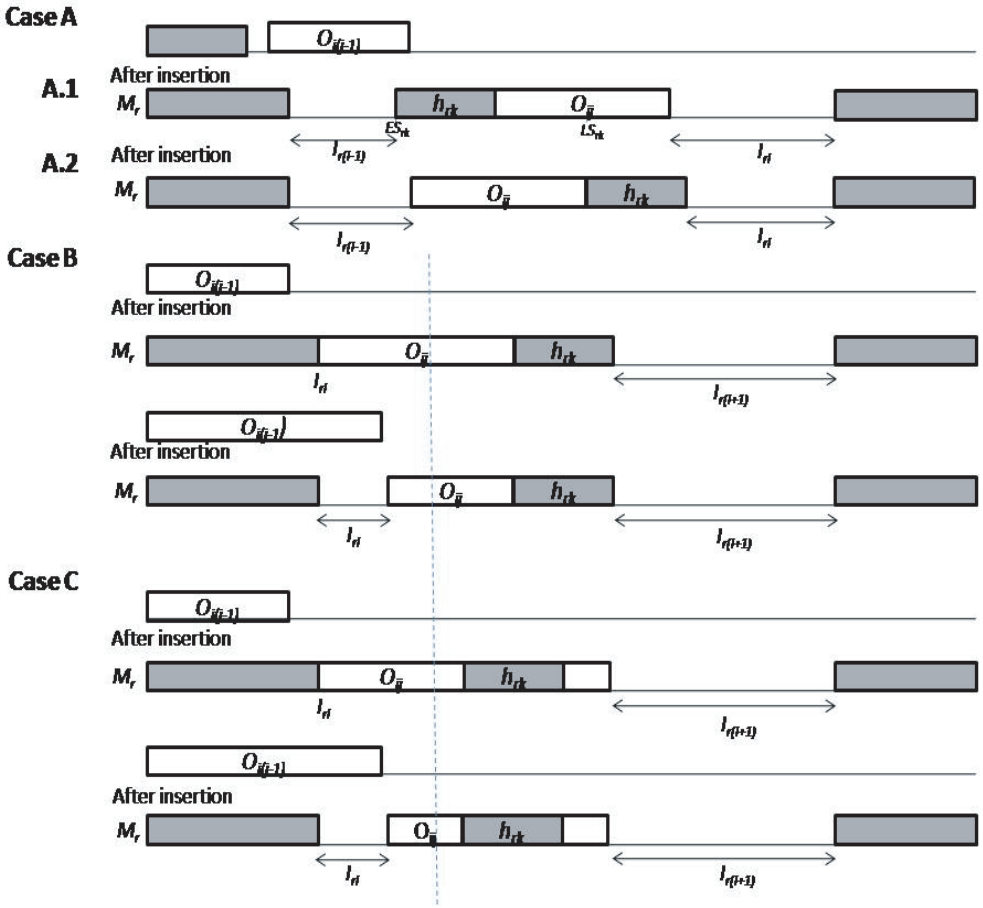

Figure 5.2: Comparison between Cases A, B and C after inserting operation O_{ij}

date ($ES_{rk} \leq \tilde{S_{rk}}$). h_{rk} cannot start before the starting date of the previous interval $I_{r(l-1)}$; otherwise it will overlap with another operation already scheduled or an unavailability period ($SI_{r(l-1)} \leq \tilde{S_{rk}}$).

The following values are calculated:

$$\tilde{S_{rk}} = max\{SI_{r(l-1)}, ES_{rk}, C_{i(j-1)} - p'_{rk}\}$$
$$t_{ij} = \tilde{S_{rk}} + p'_{rk}$$
$$C_{ij} = t_{ij} + p_{ij}$$

To entirely insert the operation in the interval by moving the unavailability period, the following condition must be satisfied:

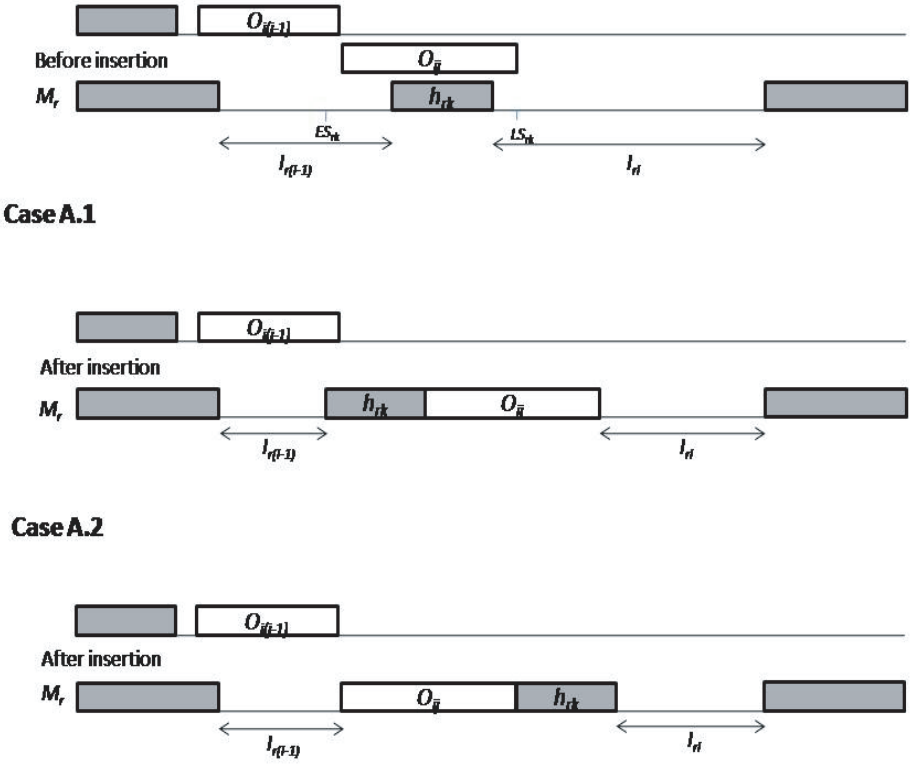

Figure 5.3: Operation insertion according to Case A

$$C_{ij} \leq TI_{rl}$$

Then the unavailability period and the intervals are updated:

$$S_{rk} = \tilde{S_{rk}}$$
$$T_{rk} = \tilde{S_{rk}} + p'_{rk}$$
$$TI_{r(l-1)} = \tilde{S_{rk}}$$
$$SI_{rl} = C_{ij}$$

If $SI_{r(l-1)} = \tilde{S_{rk}}$ (resp. $C_{ij} = TI_{rl}$), the interval $I_{r(l-1)}$ (resp. I_{rl}) is deleted.

Case A.2

- Unavailability period h_{rk} can move to the right in its time window ($S_{rk} < LS_{rk}$)

- The precedent operation in the job routing $O_{i(j-1)}$ must finish after the end date of the previous interval $I_{r(l-1)}$ ($TI_{r(l-1)} \leq C_{i(j-1)}$) otherwise it is similar to case B.

It is not necessary for unavailability period h_{rk} to move to start before the end of the current operation O_{ij} ($C_{ij} \leq \tilde{S_{rk}}$). h_{rk} cannot start after its latest starting date ($\tilde{S_{rk}} \leq LS_{rk}$). h_{rk} cannot finish after the end of the current interval I_{rl} ($\tilde{S_{rk}} \leq TI_{rl} - p'_{rk}$).

This implies that

$C_{ij} \leq \tilde{S_{rk}} \leq min\{LS_{rk}, TI_{rl} - p'_{rk}\}$

The following values are calculated:

$t_{ij} = C_{i(j-1)}$
$C_{ij} = t_{ij} + p_{ij}$

Hence, the operation can be entirely inserted in the interval by moving the unavailability period if the following condition is satisfied:

$$C_{ij} + p'_{rk} \leq min\{LS_{rk} + p'_{rk}, TI_{rl}\}$$

Then the unavailability period and the intervals will be updated as follows:

$S_{rk} = C_{ij}$
$T_{rk} = S_{rk} + p'_{rk}$
$TI_{r(l-1)} = t_{ij}$
$SI_{rl} = T_{rk}$

If $TI_{rl} = T_{rk}$, the interval I_{rl} will be deleted.

Case B: Flexibility The difference between cases A and B is that, in case A, we are interested in moving the unavailability periods placed at the beginning of interval I_{rl} and, in case B, at the end of the interval. In cases A.2 and B, the unavailability period is moved to the left. The difference is that, in case A.2, operation O_{ij} is inserted in interval $I_{r(l-1)}$ whereas, in case B it is inserted in I_{rl}.

The conditions for case B are:

- The previous operation in the job routing $O_{i(j-1)}$ must finish before the end date of the current interval I_{rl} ($C_{i(j-1)} < TI_{rl}$), otherwise it is the same as case A.2.
- The end date of the current interval I_{rl} is the starting date of unavailability period h_{rk} ($TI_{rl} = S_{rk}$)
- The starting date of the next interval $I_{r(l+1)}$ corresponds to the end date of unavailability period h_{rk} ($SI_{r(l+1)} = T_{rk}$).

- Unavailability period h_{rk} can move to the right in its time window ($S_{rk} < LS_{rk}$) (see Figure 5.4)

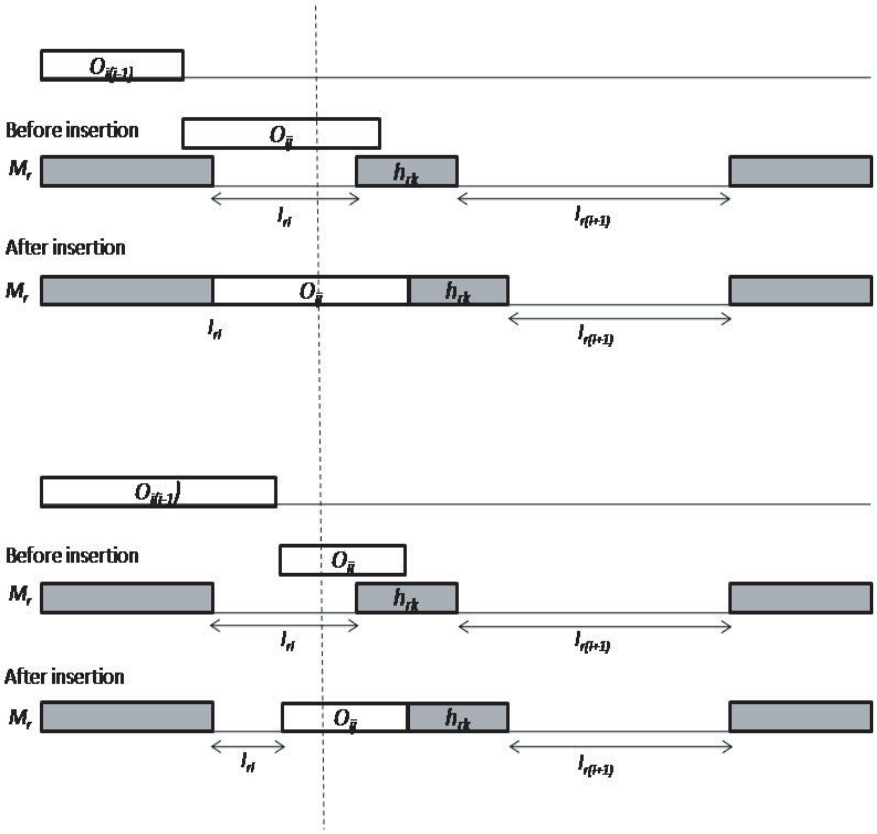

Figure 5.4: Operation insertion according to Case B.

It is not necessary for unavailability period h_{rk} to move more than the end of the current operation O_{ij} ($C_{ij} \leq \tilde{S_{rk}}$). h_{rk} cannot start after its latest starting date ($\tilde{S_{rk}} \leq LS_{rk}$). h_{rk} cannot finish after the end of the next interval ($\tilde{S_{rk}} \leq TI_{r(l+1)} - p'_{rk}$).

This implies that

$$C_{ij} \leq \tilde{S_{rk}} \leq min\{LS_{rk}, TI_{r(l+1)} - p'_{rk}\}$$

The following values are calculated:

$$t_{ij} = max\{C_{i(j-1)}, SI_{rl}\}$$
$$C_{ij} = t_{ij} + p_{ij}$$

The operation can be entirely inserted in the interval by moving the unavailability period if:

$$C_{ij} + p'_{rk} \leq min\{LS_rk + p'_{rk}, TI_{r(l+1)}\})$$

then the unavailability period and the intervals will be updated:

$$S_{rk} = C_{ij}$$
$$T_{rk} = S_{rk} + p'_{rk}$$
$$TI_{rl} = t_{ij}$$
$$SI_{r(l+1)} = T_{rk}$$

If $t_{ij} = SI_{rl}$ (resp. $TI_{r(l+1)} = T_{rk}$), the interval I_{rl} (resp. $I_{r(l+1)}$) will be deleted.

Case C: Preemption The representation of this case is the same as for case B. The only difference is that, in case C, the unavailability period cannot move. Since the preemption is allowed, operation O_{ij} is interrupted by unavailability period h_{rk} and inserted in intervals I_{rl} and $I_{r(l+1)}$.

The conditions for case C are listed below (See Figure 5.5):

- The previous operation in the job routing $O_{i(j-1)}$ must finish before the end date of the current interval I_{rl} ($C_{i(j-1)} < TI_{rl}$) (as for case B).
- The end date of the current interval I_{rl} corresponds to the starting date of unavailability period h_{rk} ($TI_{rl} = S_{rk}$).
- The starting date of the next interval $I_{r(l+1)}$ is the end date of unavailability period h_{rk} ($SI_{r(l+1)} = T_{rk}$)
- The operation cannot be entirely inserted in the current interval I_{rl} and unavailability period h_{rk} cannot move to the right in its time window ($S_{rk} = LS_{rk}$).
- The preemption is allowed $\beta_{ijk} = 1$ (h_{rk} crossable and O_{ij} preemptive).

The following values are calculated (interruption of operation O_{ij} by unavailability period h_{rk}):

$$t_{ij} = max\{SI_{rl}, C_{i(j-1)}\}$$
$$C_{ij} = t_{ij} + (S_{rk} - t_{ij}) + p'_{rk} + [p_{ij} - (S_{rk} - t_{ij})] + \alpha_{ij}(S_{rk} - t_{ij})$$

where $(S_{rk} - t_{ij})$ is the part of O_{ij} processed before the beginning of h_{rk}, $(p_{ij} - (S_{rk} - t_{ij}))$ the part remaining to carry out and $\alpha_{ij}(S_{rk} - t_{ij})$ the part to redo.

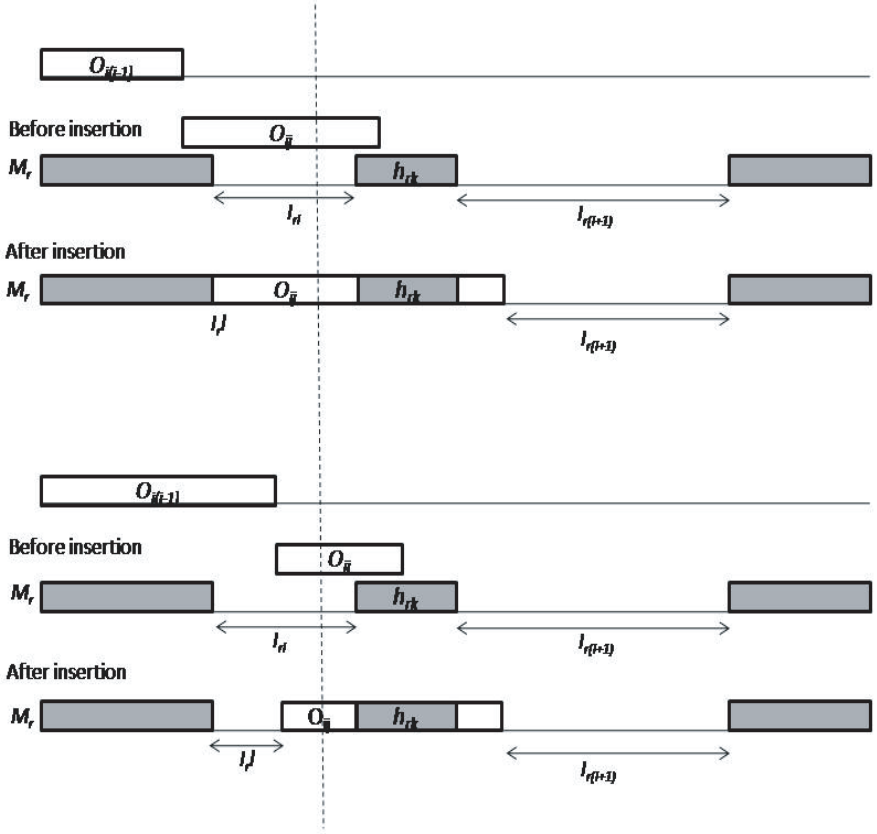

Figure 5.5: Operation insertion according to Case C.

To simplify, $C_{ij} = t_{ij} + p'_{rk} + p_{ij} + \alpha_{ij}(S_{rk} - t_{ij})$

If the operation can be entirely inserted in the interval when interrupted by the unavailability period by satisfying the condition

$$C_{ij} \leq TI_{r(l+1)},$$

then the unavailability period and the intervals will be updated:

$$TI_{rl} = t_{ij}$$
$$SI_{r(l+1)} = C_{ij}$$

If $t_{ij} = SI_{rl}$ (resp. $TI_{r(l+1)} = C_{ij}$), the interval I_{rl} (resp. $I_{r(l+1)}$) will be deleted.

5.1.2 Procedure of Operation Insertion in the interval (OIp)

This procedure describes a strategy to insert operation O_{ij} in interval I_{rl}. Five main steps are possible to achieve the insertion (See Figure 5.6):

Step 1: The operation can be entirely inserted in the interval ($p_{ij} \leq TI_{rl} - max\{C_{i(j-1)}, SI_{rl}\}$). Note that O_{ij} cannot start before $max\{C_{i(j-1)}, SI_{rl}\}$. Check if there is an unavailability period and if it can move (Case A). The objective is to insert the operation as early as possible. Else, insert the operation and update the interval. ($t_{ij} = max\{C_{i(j-1)}, SI_{rl}\}, C_{ij} = t_{ij} + p_{ij}$)

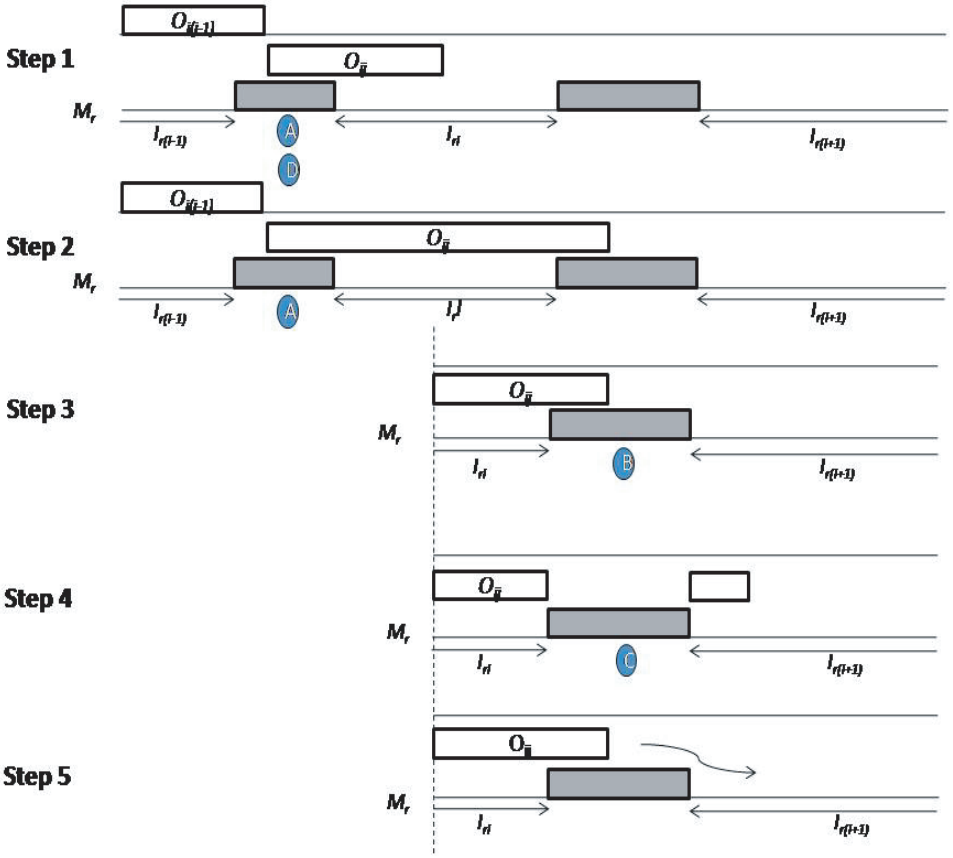

Figure 5.6: OIp procedure.

In this case, designated by Case D and illustrated in Figure 5.7,

- When $t_{ij} = SI_{rl}$ and $C_{ij} = TI_{rl}$ (resp. $C_{ij} < TI_{rl}$), the interval I_{rl} will be deleted (resp. $SI_{rl} = C_{ij}$)
- When $t_{ij} = SI_{rl}$ and $C_{ij} < TI_{rl}$, $SI_{rl} = C_{ij}$

– When $t_{ij} > SI_{rl}$ and $C_{ij} = TI_{rl}$ (resp. $C_{ij} < TI_{rl}$), $TI_{rl} = t_{ij}$ (resp. a new interval I'_{rl} will be created and the intervals will be updated as follows: $TI_{rl} = t_{ij}$, $SI'_{rl} = C_{ij}$, $TI'_{rl} = TI_{rl}$).

– When $t_{ij} > SI_{rl}$ and $C_{ij} < TI_{rl}$, a new interval I'_{rl} will be created and the intervals will be updated as follows: $TI_{rl} = t_{ij}$, $SI'_{rl} = C_{ij}$, $TI'_{rl} = TI_{rl}$.

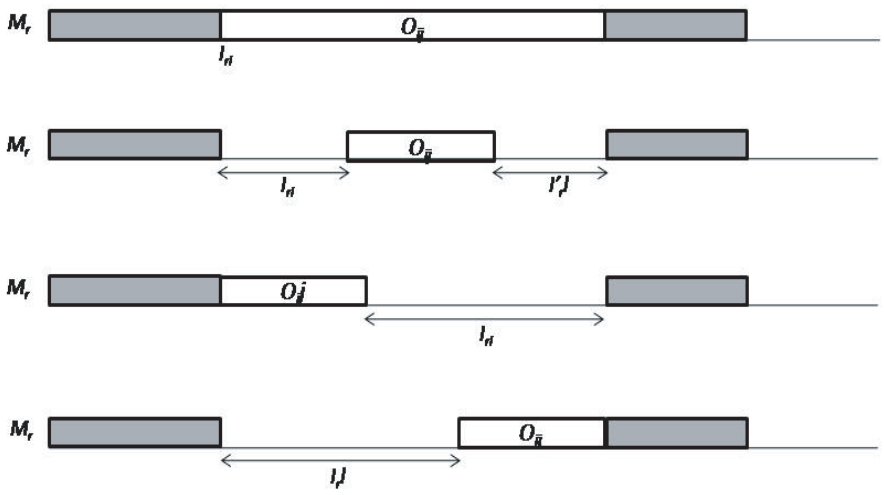

Figure 5.7: Operation insertion according to Case D.

If the operation cannot be entirely inserted in the interval ($p_{ij} > TI_{rl} - max\{C_{i(j-1)}, SI_{rl}\}$), go to Step 2.

Step 2: Check if the unavailability period can move (Case A). The objective is to insert the operation in the interval. Else, go to step 3.

Step 3: Check if there is an unavailability period and if it can move to the right (Case B). The objective is to insert the operation in the interval. Else, go to Step 4.

Step 4: Check if preemption is allowed. Check if the operation can be inserted in the current and

following intervals (Case C). The objective is to insert the operation in the interval. Else, go to Step 5.

Step 5: Search for another interval. The next one is $I_{r(l+1)}$ as we are searching to insert the operation as early as possible. Perform OIp procedure with $I_{rl} \leftarrow I_{r(l+1)}$.

Remarks:

1. *Two unavailability periods are not too close to each other (this assumption is also considered in Chapter 4). Then when inserting an operation in an interval, this operation is in presence of only one unavailability period. Hence, we have either Case A or Case (B or C).*

2. *For the first operation of the job ($j = 1$), as it has no predecessor, we set $C_{i(j-1)} = 0$. It is then only a fictive value used to keep unchange some of the formulas used above.*

3. *If the current interval I_{rl} is the first interval on the machine ($l = 1$), Case A.1 cannot be valid and, for, Case A.2 we set $SI_{r(l-1)} = TI_{r(l-1)} = 0$.*

4. *The way the cases are ordered implies that priority is given to the flexibility. We can reorder the cases starting by C and then A and B, to set the priority to preemption. Note that each time that the order changes, a new scheduling strategy is defined. In the experiments, other strategies are tested.*

5. *Procedure OIp is a building block composed of other building blocks.*

5.1.3 Job based heuristics

In the following, we present three methods that prioritize the job operations. In the first two heuristics, Job priority Heuristic (JpH) and Operation priority Heuristic 1 (OpH1), we introduce randomness in the ordering of the jobs and the operations for the insertion respectively. In the third heuristic OpH2, the operation to insert is selected according to a given rule. JpH and OpH1 can be used as evaluating blocks in methods that use many solutions at the same time to provide better ones as it is the case for genetic algorithms. OpH2 can be used to provide a starting solution for methods that, from an initial solution, try to improve the obtained solution at each iteration. The solution is evaluated at each iteration by JpH or OpH1 (depending on the structure of the priority sequence).

5.1.3.1 Job priority Heuristic (JpH)

This heuristic was proposed by Aggoune [Agg02] for the non-preemptive job shop scheduling problem with machine availability constraints. We introduce a new procedure for operation insertion to take into account the flexibility on starting dates of the unavailability periods and operation preemption, when inserting an operation in an interval. The flexibility was treated

in [Agg02] by positioning each unavailability period at the end of its time window. Then, in the resulting schedule an operation or an unavailability period are moved to start earlier each time that it is possible.

The construction of a schedule is made job after job. To construct a feasible schedule, the heuristic operates according to the following steps: Use an initial priority sequence s between the jobs $\{J_1, .., J_n\}$. Each time a job J_i is selected, the heuristic determines, for each operation O_{ij}, with respect to its order in the job routing, the subinterval in the planing horizon on the associated machine mr_{ij} to be processed as early as possible, without overlapping an unavailability period or an operation already scheduled. After each insertion, intervals and unavailability periods are updated. A new interval may be created or an existing one can be deleted. The main steps are summarized in Algorithm 4

The operation insertion in the interval is made according to the Operation Insertion procedure (OIp).

Note that the performance of the heuristic strongly depends on the job priority order in the initial sequence. This is why, in the experiments, the heuristic is performed several times in order to evaluate its performance. To obtain better results, it can be combined with a metaheuristic.

Algorithm 4 Algorithm of Job priority Heuristic (JpH)

Begin
  Define an initial job priority sequence s
  for each job J_i in the sequence s **do**
    for each operation O_{ij} in the job routing **do**
      Use the procedure OIp to insert operation O_{ij} on its machine m_{ij}
    end for
  end for
end

Remarks:

- JpH heuristic is designed according to the following rule:
  Rule (1). Scheduling all the operations of a job before moving to the next one.
  In the sequel, when using JpH, it means that it concerns Rule (1).

- Using the following rule defines another way to schedule the operations according to the initial job sequence:
  Rule (2). For $j = 1$ to $\max_{1 \leq i \leq n} n_i$, j^{th} operations in the routings of the jobs are ordered in respect to sequence s.
  We refer to the job priority heuristic using Rule (2) as **JpH-R2**

5.1.3.2 Operation priority Heuristic 1 (OpH1)

It is quite similar to the JpH heuristic. The difference is that the priority is set for operations instead of jobs. It allows to explore solution configurations that cannot be obtained by JpH. Experiment results show that it is relevant for C_{max} but not for $\sum_i C_i$ as the results are better by JpH.

The construction of a schedule is made one operation after the other. To construct a feasible schedule, the heuristic operates according to the following main steps (see Algorithm 5): Use an initial priority sequence s for operations $(O_1, .., O_{no})$, where $no = \sum_{i=1}^{n} n_i$ is the total number of operations. The operations are successively inserted as early as possible with the OIp procedure and the intervals updated after each insertion.

Note that, as for JpH, the performance of the heuristic strongly depends on the operation priority order in the initial sequence. Better results can be obtained by combining with another method.

Algorithm 5 Algorithm of Operation priority Heuristic 1 (OpH1)

```
Begin
  Find an initial operation priority sequence s
    for each operation O in the sequence s
      Use the procedure OIp to insert operation O on its machine
    end for
end
```

In the operation sequence, operations that belong to the same job are denoted by the same symbol which is the job number. To know the position of an operation in a job, we check its occurrence in the sequence. Each job J_i appears exactly ni times. This sequence coding is proposed by Gen *et al.* [GTK94]. for solving the job-shop problem using genetic algorithm. It is also used by Zribi [Zri05] in a genetic algorithm for the flexible job shop problem.

5.1.3.3 Operation priority Heuristic 2 (OpH2)

As for OpH1, the priority is set on operations instead of jobs. The difference between OpH1 and OpH2 consists in using a rule for selecting an operation; and not an ordered sequence of operations defined a priori.

The heuristic we propose was inspired by the one presented in Esquirol and Lopez [EL99] for the classical job shop problem (non preemptive operations and continuously available machines).

In Esquirol and Lopez [EL99], the heuristic is as follows:

The construction of a schedule is made one operation after the other. To construct a feasible schedule, the heuristic operates with the following steps until all operations are scheduled:

A list O of *ready to schedule* operations is established (an operation is ready to schedule if the immediate previous operation in the job routing is completed). At the beginning, the list is composed of the first operations of all jobs.

To select an operation from that list, one of the well-known following rules is chosen:

a Select the operation which earliest starting date is the smallest: $min_{O_{ij} \in O} Et_{ij}$

b Select the operation whose earliest completion date is the smallest: $min_{O_{ij} \in O} EC_{ij}$.

The rule (a) generates non-delay schedules; i.e. are schedules in which a machine does not remain idle if an operation is waiting to be processed on it. However, there is no guarantee that the set of non-delay schedules contains the optimal solution (Esquirol and Lopez [EL99]). This is not the case for the active schedules constructed by rule (b); that are schedules such as it is impossible to move an operation without delaying the start time of another operation.

The earliest starting date Et_{ij} is given by the maximum between the completion date $C_{i(j-1)}$ of the predecessor of O_{ij} in the job routing and the end of the last operation on the machine that must process O_{ij} that we denote by $ready_r$ for ready date of the machine $r = mr_{ij}$. Then $Et_{ij} = max\{C_{i(j-1)}, ready_r\}$.

The earliest completion date EC_{ij} corresponds to the earliest starting date Et_{ij} plus the processing time p_{ij} ($EC_{ij} = Et_{ij} + p_{ij}$). On the machine on which the operation is to be processed, the set of ready to schedule operations O_t^* is composed of operations O_{ij*} such that $Et_{ij*} < min_{O_{ij} \in O} EC_{ij}$.

Hence, from that list, the operation which satisfies the chosen rule is selected. If there are more than one operation, to select one of them, the priority rule MWKR (Most WorK Remaining) is used. This rule gives the priority to the operation of the job for which the remaining work to process is the biggest. The aim is to balance the processing of the jobs corresponding to those operations. So, for each operation, the total processing time of the remaining operations of the corresponding job, including of course the operation, is calculated. Then, the resulting processing times are compared; and the operation corresponding to the maximum value is selected.

Note that, at each iteration, the list of ready to schedule operations is almost the same as the previous. Only one operation changes. It is replaced by the successor operation in the job routing. In addition, the values of the parameter of the chosen rule remain the same except for operations which are to be processed on the same machine than the selected operation.

As we are dealing with a problem with non continuously available machines, flexible starting dates of unavailability periods and preemption between operations and unavailability periods, the earliest starting date of an operation may not satisfy $Et_{ij} = max\{C_{i(j-1)}, ready_r\}$ because of the presence of unavailability periods on the machine. Indeed, the machine is available by intervals and the interval corresponding to time $max\{C_{i(j-1)}, ready_r\}$ may not be sufficiently large to contain operation O_{ij}. Moreover, $EC_{ij} = Et_{ij} + p_{ij}$ is not systematically satisfied. This is due to the fact that, in case of unavailability on the machine, the completion date may also include the duration of an unavailability period and a penalty on preemption; that is $EC_{ij} = Et_{ij} + p'_{rk} + p_{ij} + \alpha_{ij}(S_{rk} - Et_{ij})$ where p'_{rk} is the duration of unavailability period h_{rk} that interrupts O_{ij} and α_{ij} the preemption penalty of O_{ij} (refer to Case C).

Let us denote by aEt_{ij}: the starting date of the first availability interval on the machine after the completion time $C_{i(j-1)}$ of the previous operation of job J_i. Note that Et_{ij} and EC_{ij} are determined by the simulation of the processing of OIp procedure; where Et_{ij} and EC_{ij} replace t_{ij} and C_{ij}, the operation is not inserted and the availability intervals and unavailability periods are not updated.

The rules that we use to select an operation from the list of ready to schedule operations are:

Rule (1). Select the operation whose starting date is the smallest: $min_{O_{ij} \in O} Et_{ij}$.

Rule (2). Select the operation whose completion date is the smallest: $min_{O_{ij} \in O} EC_{ij}$.

Rule (3). Select the operation for which the corresponding idle time on the machine is the smallest and is larger than or equal to 0: $min_{O_{ij} \in O} idle_{ij}$.

Rule (4). the operation which the corresponding beginning idle time on the machine is the smallest and is larger than or equal to 0: $min_{O_{ij} \in O} idle1_{ij}$. If this condition is satisfied by more than one operation, rule (3) is used.

Rule (5). Select the operation whose starting date of the first availability interval on the machine after the completion of the previous operation in the job routing: $min_{O_{ij} \in O} aEt_{ij}$.

Rule (6). Select the operation corresponding to the maximum tail which represents the total processing time of the remaining operations to process on the machine: $max \sum_{mr_{i'j'} = mr_{ij}} p_{i'j'}$.

Rule (1) is the same as Rule (a). The disadvantage of Rule (5) is that if the selected operation O_{ij} cannot be inserted in the corresponding availability interval, it may be inserted in an interval that is far from aEt_{ij} date. The aim of Rule (6) is to balance the workload on the machines.

As minimizing the makespan is equivalent to minimizing the total idle time on the machines, Rules (3) and (4) are used. However, for Rule (3), by setting the choice to the smallest positive idle time, Case C will not operate, and the preemption is no longer considered; which is not

the case for Rule (4) except when more than one operation satisfy the condition. The idle time $idle_{ij}$ associated to an operation O_{ij} is decomposed into an idle time $idle1_{ij}$ before the beginning of operation and an idle time $idle2_{ij}$ at the end of operation. To compare two operations O_{ij} and $O_{i'j'}$ we use the following condition:

O_{ij} is preferred to $O_{i'j'}$ if $0 \leq idle_{ij} \leq idle_{i'j'}$.

In some cases, it is interesting to consider the condition $idle1_{ij} \leq idle1_{i'j'}$ to minimize the idle time before the beginning of the inserted operation. This condition is interesting in case where this idle time may be not sufficiently large to contain an operation. However, in case where this idle time is sufficiently large, the condition is not relevant.

To obtain the values of the tail, and the idle times, we simulate the processing of OIp procedure. No updating of the starting and completion dates of operations, the unavailability period and/or the availability intervals is performed.

Once the operation is selected, it is then inserted in the interval using OIp.

The main steps of the heuristic are described in Algorithm 6

Algorithm 6 Algorithm of Operation priority Heuristic 2 (OpH2)

Begin
  Repeat
    Define the list of operations ready to schedule
    Select the operation which satisfies the chosen rule
    If there are more than one operation
      Apply the MWKR rule
    end if
    Insert the selected operation on the associated machine using OIp procedure
    Remove the operation from the list of operations remaining to schedule
  Until the list of operations to schedule is empty
end

5.1.4 Machine based Heuristics

The principle of the following heuristics is the same as for list algorithms: the priorities are associated to machines. And it is the choice of the machine that defines the set of ready to schedule operations to select.

Indeed, at first the *ready machine* must be defined. It corresponds to the machine which is ready earlier. The selection of this machine is done according to:

1. For each machine that has ready to schedule operations to process on it, define the *ready date*. It corresponds to the *first availability* date on the machine.

2. The ready machine corresponds to the one whose ready date is the smallest.

In absence of unavailability periods, the ready date of the machine is defined by the last operation processed on it. However, when there are unavailability periods, the machine can have availability intervals that are prior to the last operation processed on it. In addition, considering the first availability interval as the ready machine, can lead to a very small interval that cannot contain any operation. Hence this machine can be selected several times, the operation is then inserted very far from that interval before moving to another machine. In the heuristics, we use the ready date as the completion date of the last operation processed on the machine.

5.1.4.1 Machine-Operation priority Heuristic 1 (MOpH1)

For this heuristic their are two priorities: machine and operation. The first priority is given to machines. The operations are inserted one by one. Each time, before the selection of any operation, the ready machine must first be defined.

As described in Algorithm 7, given an initial operation sequence, the heuristic operates as follows: the ready machine is first defined then, from the set of the ready to schedule operation to process on that machine, select the first one in the sequence of operation priorities. The selected operation will be inserted in the appropriate availability interval defined by OIp procedure; and the first availability of the machine will be updated.

Algorithm 7 Algorithm of Machine-Operation priority Heuristic 1 (MOpH1)

```
Begin
    Define an initial operation priority sequence s
        while operations remain to schedule
            Define the ready machine
            Select the most prior operation O ready to schedule to process on the
            ready machine
            Use the procedure OIp to insert operation O on its machine
            Update the availability date of the machine with the completion date of
            the scheduled operation
            Remove the operation from the list of operations remaining to schedule
        end while
end
```

5.1.4.2 Machine-Operation priority Heuristic 2 (MOpH2)

This method is a combination of OpH2 and MOpH1. As for MOpH1, the ready machine must be defined at the beginning of each iteration. Then, as for OpH2, the operation corresponding to the chosen rule (from (1) to (6)), is selected from the set of the ready to schedule operations to process on the ready machine. Algorithm 8 summarizes the main steps of the method.

Algorithm 8 Algorithm of Machine-Operation priority Heuristic 2 (MOpH2)

```
Begin
    Repeat
        Define the ready machine
        Define the list of operations ready to schedule on this machine
        Select the operation which satisfies the chosen rule
        If there are more than one operation
            Apply the MWKR rule
        end if
        Insert the selected operation on the associated machine
        Update the availability date of the machine with the completion date of the
        scheduled operation
        Remove the operation from the list of operations remaining to schedule
    Until the list of operation to schedule is empty
end
```

5.1.5 Implementation and experimentation

The tests are performed on a set of benchmarks that are composed by the five instances of (number of machines x number of jobs) groups $(5, 5)$, $(5, 10)$, $(10, 10)$, $(10, 15)$ used in Chapter 4.

Concerning the JpH, OpH1 and MOpH1 heuristics, as their performances depend strongly on the priorities defined by the sequences, they are performed several times Although, the heuristics can be performed with no limit on the number of iterations, we choose to present the results for number of iterations equal to 100, 1000, 10000, 100000. Therefore, the higher is the number of iterations, the higher is the computation time. Then the limit on the number of iteration can be defined by the limit time allowed to the mixed integer programming problems (MIP) resolution.

Starting from the sequence representing an order of the jobs from 1 to n, we observed that the more randomness is introduced in the generation process of the sequences, the best are the results. The sequences are generated such as a fixed number of permutations are operated to the previous ones.

For OpH2 and MOpH2, as the order of the operations is defined by rules, the sequence is the same from an iteration to another, we need then to run the experiments only once. The performances of the heuristics are studied over the six rules introduced in the theoretical part.

The heuristics are compared to the optimal solutions or lower bounds provided by the mathematical modeling.

5.1.5.1 Definition of Tables structure

As the same structure is used for some tables in Sections 5.1.5.2 and 5.1.5.3, we prefer to define all the tables in this section.

Tables 5.1 and 5.12 give respectively the ranking of the priority rules for OpH2 and MOpH2 heuristics by objective criteria C_{max} and $\sum C_i$, and fixed or flexible unavailability periods whatever is the operations character (non-preemptive, resumable, non-resumable, semi-resumable). This ranking is defined according the number of the best and worst solutions the rules determine overall the instances. Column 1 represents the rank. Columns 2 and 3 (resp. 4 and 5) represent the ranks for C_{max} (resp. $\sum C_i$). Columns 2 and 4 concern fixed unavailability periods; whereas Columns 3 and 5 concern flexible unavailability periods. This ranking is established thanks to Tables from A.7 to A.14 of Appendix.

Tables 5.4, 5.6, 5.8, 5.13 and 5.15 (resp.5.5, 5.7, 5.9, 5.14 and 5.16) summarize the test results for fixed (resp. flexible) unavailability periods with respectively JpH, OpH1, OpH2, MOpH1, MOpH2 heuristics. For JpH, OpH1 and MOpH1, the results concern the case of 1000 iterations. For OpH2 and MOpH2, the results gather the best values of the objective criteria through all the rules. Table 5.2 (resp. 5.3) represents the test results form Integer resolution of the disjunctive model with fixed (resp. flexible) unavailability periods. Column 1 is the name of the instance; Columns 2-3, 4-5, 6-7, 8-9 represent respectively the test results of the heuristic for non-preemptive, resumable, non-resumable and semi-resumable operations. Columns 2, 4, 6, 8 correspond to the makespan value of the solution and Columns 3, 5, 7, 9 correspond to the value of the sum of the completion dates of the jobs of the solution. For all these tables, except those associated to MIP resolution, indications on the gap between the best solution of the heuristics and the disjunctive MIP solution are given.

Table 5.11 gives the test results for resumable operations and flexible unavailability periods for orders ABC and CAB in OIp procedure with OpH2 heuristic and through all the rules. Column 1 represent the name of the instance. Columns from 2 to 5 represent the test results for OpH2 heuristics with different orders of cases A, B and C in OIp procedure, Columns 2 and 3 for ABC order and Columns 4 and 5 for CAB order. The results represent the best solutions through all the rules. Columns 6 and 7 represent the test results for the disjunctive MIP model. Columns 2, 4, 6 give the results for C_{max} and Columns 3, 5, 7 give the results for $\sum C_i$. Except for the results associated to MIP resolution, indications on the gap between the best solution of the heuristic and the disjunctive MIP solution are given.

Tables 5.17 and 5.18 gather the test results for respectively non-preemptive operations and fixed unavailability periods, and resumable operation and flexible unavailability periods for all the heuristics to compare them. Hence, Column 1 is the name of the instance; Columns 2-3, 4-5, 6-7, 8-9, 10-11 represent respectively the test results of JpH, OpH1, OpH2, MOpH1, MOpH2; and Columns 12-13 give the test results for the integer resolution of the disjunctive MIP model. Columns 2, 4, 6, 8, 10, 12 correspond to the makespan value of the solution; and Columns 3, 5, 7, 9, 11, 13 correspond to the value of the sum of the completion dates of the jobs of the solution. We highlight, for each objective criteria, the best and the worst results; and for the best results, we show the gap compared to the solution of the disjunctive MIP resolution.

Table 5.20 presents the test results of the comparison between JpH and JpH-R2 in case of 100000 iterations. Column 1 is the name of the instance; Columns from 2 to 7 (resp. from 8 to 13) represent the results in case of non-preemptive (resp. resumable) operations and fixed (resp. flexible) unavailability periods. Columns 2-3, 8-9 concern JpH. Columns 4-5, 10-11 concern JpH-R2. Columns 6-7, 12-13 concern disjunctive MIP. Columns 2, 4, 6, 8, 10, 12 correspond to the makespan value of the solution; and Columns 3, 5, 7, 9, 11, 13 correspond to the value of the sum of the completion dates of the jobs of the solution. For each objective criteria, we highlight the best results.

Table 5.21 presents the test results comparing JpH-R2 and the best results over all the heuristics (including JpH). Column 1 is the name of the instance; Columns from 2 to 7 (resp. from 8 to 13) represent the results in case of non-preemptive (resp. resumable) operations and fixed (resp. flexible) unavailability periods. Columns 2-3, 8-9 represent the best results over all the heuristics (including JpH). Columns 4-5, 10-11 represent the results for JpH-R2. Columns 6-7, 12-13 represent the results for disjunctive MIP. Columns 2, 4, 6, 8, 10, 12 correspond to the makespan value of the solution; and Columns 3, 5, 7, 9, 11, 13 correspond to the value of the sum of the completion dates of the jobs of the solution. For each objective criteria, we highlight the best results.

Note that, although the heuristics integrates all the availability models, we present sometimes the test results only for the less flexible (non-preemptive operations and fixed unavailability periods) and the most flexible (resumable operations and flexible unavailability periods) cases.

5.1.5.2 Job based heuristics

Test results (See Tables from A.1 to A.4 of Appendix) show that, for heuristics JpH and OpH1, the higher is the number of iterations the best are the solutions. Indeed, the results are generally improved through number of iterations of 100, 1000, 10000 and 100000; and the best results are obtained for 100000 iterations. However, for instances of class (5,5), the results are stable and can be optimal; and for the case of fixed unavailability periods, most of them are

optimal. In addition, for few benchmarks, the best solution is obtained for number of iterations lower than 100000.

The best results for the heuristics are either equal to those of the disjunctive MIP model or the gap between these values and those of the MIP model is less or equal to 10%. There are also few benchmarks for which the objective function values are better than those of the MIP model. Of course the remaining values of the heuristics are worse than the ones of the MIP model.

Concerning criterion C_{max}, the best results are, in general, given by OpH1 heuristic. Whereas, for criterion $\sum C_i$, the best results are obtained by JpH heuristic.

Test results (see Tables A.5 and A.6 of Appendix) show that the higher is the number of iterations, the higher is the CPU time. Moreover, the CPU time of each level is approximatively equal to the CPU time of the lower level multiplied by 10; note that the number of iterations of the level is equal to the number of iterations of the lower level multiplied by 10. The case of resumable operations and flexible unavailability periods induces a higher CPU time than the case of non-preemptive operations and fixed unavailability periods. Moreover, the CPU time that is required to solve the instances by JpH heuristic is slightly lower that the CPU time for OpH1 heuristic. Except for instances (5,5) for number of iterations of 10000 or 100000, the CPU times associated to JpH and OpH1 are better than those of MIP model.

For each rule, the CPU time for constructing a schedule and calculating C_{max} and $\sum C_i$ values are ≤ 0.02 seconds.

Table 5.1: Rules rank for OpH2 and all availability models.

Rank	C_{max}		$\sum C_i$	
	Fixed unavailability periods	Flexible unavailability periods	Fixed unavailability periods	Flexible unavailability periods
1	1	2	1	2
2	5	1	5 or 6	4
3	6	4	5 or 6	1
4	4	3	2 or 4	5 or 6
5	2 or 3	6	3	5 or 6
6	2 or 3	5	2 or 4	3

Rules ranking is established relatively to the amounts of best and worst solutions obtained by each heuristic over non-preemptive, resumable, non-resumable and semi-resumable problems in case of fixed or flexible unavailability periods for C_{max} and $\sum C_i$. An aggregation of the ranks between non-preemptive, resumable, non-resumable and semi-resumable allows to establish the

global rank by fixe and flexible characters of unavailability periods.

Table 5.1 shows that the best rule for C_{max} is Rule (1); whereas the best one for $\sum C_i$ is Rule (2); almost best solutions are obtained thanks to these rules. The worst rule for C_{max} is Rule (2) or (3) (resp. (2) or (4)) depending on the operations preemptive character and fixed (resp. flexible) unavailability periods; whereas the worst one for $\sum C_i$ is Rule (5) (resp. (3)) for fixed (resp. flexible) unavailability periods. Recall that Rules (3) and (4) deal with idle times; we observe that considering the minimum idle time before the beginning of operations eligible to insertion allows to find better results than when the total idle time is considered for each operation. The reasons may be that trying to insert the operation as early as possible in the insertion interval may create an idle time after the completion of the operation sufficiently large to insert another operation.

Table 5.2: Test results for fixed unavailability periods with disjunctive MIP model.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	895*	3652*	845*	3577*	895*	3652*	866.62*	3613.00*
5m5n2	1096*	4669*	997*	4385*	1096*	4669*	1042.25*	4601.00*
5m5n3	1070*	4008*	1020*	3912*	1070*	4008*	1046.50*	3938.00*
5m5n4	1147*	4584*	1135*	4136*	1147*	4584*	1137.00*	4307.00*
5m5n5	1202*	4663*	1108*	4443*	1202*	4663*	1159.50*	4597.50*
5m10n1	1361*	9850*	1322	9610	1361	9850	1378.50	9843.00
5m10n2	1455	10637*	1400	10402	1455	10643	1452.75	10440.50
5m10n3	1607	11072	1583	11130	1637	11072	1625.50	11106.00
5m10n4	1537*	10630*	1492	10526	1537	10630	1503.50	10588.00
5m10n5	1415	9706*	1386	9407	1415	9706	1406.00	9515.00
10m10n1	1948	16878	1907	16335	1987	17117	1906.00	16806.00
10m10n2	1917*	15777*	1822	15353	1987	17117	1957.50	15572.70
10m10n3	1875	15352*	1815	15349	1875	15644	1825.25	15352.00
10m10n4	1772*	15374*	1755	15423	1772	15374	1780.00	15411.20
10m10n5	1975	15625*	1870	15669	1933	15719	1938.37	15595.00
10m15n1	2089	27199	2310	27010	2209	26886	2107.00	26993.20
10m15n2	2316	29186	2460	29251	2326	u	2366.00	u
10m15n3	2261	u	2313	27196	2213	28308	2271.00	u
10m15n4	2254	27423	6958	27530	2212	27763	2214.00	27277.20
10m15n5	2282	28609	6392	27601	2342	19226.97	2272.25	28805.20

*: optimal solution

u: unknown solution

Tables 5.2, 5.4, 5.6 and 5.8 show that dominance between non-preemptive, resumable, non-resumable and semi-resumable problems in case of fixed unavailability periods are conserved by JpH, OpH1 and OpH2 for the two objective criteria C_{max} and $\sum C_i$. The results for the resumable case are better than those for non-preemptive case. The results for the non-preemptive

Table 5.3: Test results for flexible unavailability periods with disjunctive MIP model.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	825*	3552*	825*	3356*	825*	3352*	825.00*	3366.50*
5m5n2	1076*	4412*	977*	4174*	1076*	4412*	1000.00*	4218.25*
5m5n3	1034*	3968*	1020*	3892*	1034*	3968*	1031.50*	3918.00*
5m5n4	1108*	4242*	1108*	4136*	1108*	4242*	1108.00*	4220.50*
5m5n5	1182*	4565*	1108*	4417*	1182*	4565*	1145.00*	4528.00
5m10n1	1300*	9535*	1316	9484	1302	9585	1339.25	9535.00
5m10n2	1400	10256*	1400	10364	1400	10274	1400.00	10274.00
5m10n3	1578	10997	1563	10710	1578	10558	1578.00	10909.00
5m10n4	1492	10491*	1492	10439	1492	10493	1496.50	10493.00
5m10n5	1372*	9101*	1372	9101	1377	9292	1372.00	9101.00
10m10n1	1859	16085	1912	16397	1912	16297	1920.00	16425.00
10m10n2	1839*	15200*	1915	15190	1874	15213	1864.62	15213.00
10m10n3	1773*	15164	1726	15261	1773	15278	1809.50	15266.00
10m10n4	1690*	15155	1701	14876	1690	15045	1714.00	15166.50
10m10n5	1833	15136	1839	15273	1880	15347	1871.00	15219.00
10m15n1	2103	27221	2087	26762	2137	u	2199.00	28108.00
10m15n2	2388	27977	2450	28685	2395	28004	2413.00	28712.50
10m15n3	2221	u	4202	27221	2403	27669	2269.00	26776.50
10m15n4	2120	27143	4941	27264	2166	28015	2562.00	26108.00
10m15n5	2225	28312	2292	28000	2146	27763	2183.00	27804.00

*: optimal solution

u: unknown solution

and non-resumable cases are close. The results for the semi-resumable case are bad comparing to those for the resumable case and better than those of the non-preemptive and non-resumable cases. OpH1 has better dominance than JpH. OpH1 provide better results for C_{max} and JpH provide better results for $\sum C_i$. Although, better solutions are found with higher number of iterations, we choose to present the results for 1000 iterations.
Although almost values of each objective criterion are similar for non-preemptive, resumable, non-resumable and semi-resumable problems in case of flexible unavailability periods, we present all the cases in Tables 5.3, 5.5, 5.7 and 5.9 to show the gap between these values and the MIP solutions. The redundancy of values can be explained by the fact that the order of Cases A, B, C used in OIp procedure favors the flexibility of unavailability periods not preemption; recall that, unlike MIP model, procedure OIp does not deal with flexibility and preemption at the same time. For this reason, less optimal solutions are found by the heuristics comparing to the case of non-preemptive operations and fixed unavailability periods.

From test results (see Table 5.10 Tables A.12, A.15, A.16 of Appendix), we deduce that, except for some instances, the best results are obtained when the unavailability periods are

Table 5.4: Test results for fixed unavailability periods with JpH heuristic and 1000 iterations.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	961∼	3748∼	891∼	3665∼	985∼	3748∼	925∼	3665∼
5m5n2	1112∼	4669*	1080∼	4438∼	1112∼	4669*	1096∼	4601*
5m5n3	1157∼	4008*	1152	3912*	1157∼	4008*	1154.5∼	3938*
5m5n4	1147*	4584*	1147*	4314∼	1147*	4584*	1147∼	4421∼
5m5n5	1202*	4663*	1166∼	4526∼	1280∼	4868∼	1222∼	4794∼
5m10n1	1590	10891	1470	10442∼	1562	10924	1525	10759.5∼
5m10n2	1521	11404∼	1602	11102∼	1641	11404∼	1606.25	11328∼
5m10n3	1754∼	12087∼	1777	11871∼	1777∼	12221∼	1737∼	12060.5∼
5m10n4	1667∼	11788	1595∼	11557∼	1728	11822	1681	11819
5m10n5	1555∼	10816	1510∼	10386∼	1560∼	10909	1560	10620
10m10n1	2421	18089∼	2299	17761∼	2397	18130∼	2382	18056.62
10m10n2	2252	17430∼	2179	17021	2332	17430∼	2311	17365.5
10m10n3	2158	16739∼	2125	16584∼	2158	16739∼	2158	16739∼
10m10n4	2129	16855∼	1948	16260∼	2171	17037	2154	16388.25∼
10m10n5	2151∼	16861∼	2110	16359∼	2151	16861∼	2140∼	16370.5∼
10m15n1	2739	29088∼	2740	29520∼	2753	30387	2764	30031.75
10m15n2	2873	31784∼	2855	31086∼	2945	31931+	2915	31729+
10m15n3	2713	29816+	2646	29177∼	2713	29921∼	2671.5	29813.75+
10m15n4	2724	30008∼	2715+	29176∼	2764	30008∼	2744	29611.5∼
10m15n5	2802	31371∼	2791+	30096∼	2802	31506	2850.93	31103.5∼

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

placed at the beginning of their time windows. In general, the best values through all the rules when the unavailability periods are still placed in their initial position are the same as those of the case of the unavailability periods placed at the end of their time windows. Note that Table 5.10 gathers the best values through all the rules of the three other tables. When the unavailability periods are placed at the beginning (resp. end) of their time windows, the unavailability periods cannot start earlier (resp. later) so Case A.1 (resp. Cases A.2 and B) of OIp procedure is (resp. are) not valid.

Table 5.11 (see also Tables A.12 and A.17 of Appendix) show that using order A,B,C in OIp procedure provides better results than order C,A,B. The first order sets the priority to flexibility; whereas preemption is preferred in the second one although no penalty is induced as we consider the resumable problem.

Table 5.5: Test results for flexible unavailability periods with JpH heuristic and 1000 iterations.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	825*	3597∼	825*	3597∼	825*	3597∼	825*	3597∼
5m5n2	1076*	4468∼	1076∼	4468∼	1076*	4468∼	1076∼	4468∼
5m5n3	1137∼	3968∼	1137∼	3968∼	1137∼	3968∼	1137∼	3968∼
5m5n4	1108*	4242∼	1108∼	4242∼	1108*	4242*	1108*	4242∼
5m5n5	1190∼	4646∼	1190∼	4646∼	1190∼	4646∼	1190∼	4646∼
5m10n1	1500	10582	1500	10490	1521	10695	1515	10588
5m10n2	1501∼	10937∼	1501∼	10937∼	1501∼	10937∼	1501∼	10937∼
5m10n3	1757	11938∼	1757	11938	1757	11938	1757	11938∼
5m10n4	1698	11596	1698	11596	1698	11596	1698	11596
5m10n5	1529	10101	1529	10101	1529	10101∼	1529	10101
10m10n1	2280	17794	2291	17794∼	2280	17794∼	2280	17794∼
10m10n2	2255	16680∼	2255	16680∼	2255	16680∼	2255	16680∼
10m10n3	2118	16661∼	2118	16661∼	2118	16661∼	2118	16661∼
10m10n4	2031	16181∼	2031	16181∼	2031	16181∼	2031	16181∼
10m10n5	2055	16534∼	2055	16534∼	2055∼	16534∼	2055∼	16534∼
10m15n1	2737	28996∼	2737	28996∼	2737	28996+	2737	28996∼
10m15n2	2854	31007	2854	31007∼	2450∼	28685∼	2854	31007∼
10m15n3	2629	29338+	2629+	29338∼	2629∼	29338∼	2629	29338∼
10m15n4	2664	28914∼	2664+	28914∼	2664	28914∼	2664∼	28914
10m15n5	2765	29332∼	2765	29660∼	2765	29678∼	2765	29660∼

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

5.1.5.3 Machine based heuristics

Test results (See Tables from A.18 and A.19 of Appendix) show that, for heuristic MOpH1 as for JpH and OpH1, the higher is the number of iterations the best are the solutions. Indeed, the results are generally improved through number of iterations.

Tests results (see Table A.20 of Appendix) shows, as for JpH and OpH1, that the higher is the number of iterations, the higher is the CPU time. Moreover, the CPU time of each level is approximatively equal to the CPU time of the lower level multiplied by 10; note that the number of iterations of the level is equal to the number of iterations of the lower level multiplied by 10. The case of resumable operations and flexible unavailability periods induces a higher CPU time than the case of non-preemptive operations and fixed unavailability periods for instances of classes (5,5) and (5,10); it is the opposite for higher size of instances. Moreover, the CPU time that is required to solve the instances by MOpH1 heuristic is widely higher than the CPU time for OpH1 heuristic.

Table 5.6: Test results for fixed unavailability periods with OpH1 heuristic and 1000 iterations.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	895*	3671∼	845*	3620∼	895*	3671∼	866.62*	3653.75∼
5m5n2	1096*	4706∼	997*	4385*	1096*	4706∼	1042.25*	4604.5∼
5m5n3	1070*	4188∼	1020*	4082∼	1070*	4188∼	1046.5*	4162∼
5m5n4	1147*	4584*	1136∼	4136*	1147*	4584*	1137*	4307*
5m5n5	1202*	4725∼	1108*	4447∼	1202*	4743∼	1159.5*	4608.5∼
5m10n1	1420∼	11288	1410∼	11111	1462∼	11567	1469.75∼	11382.87
5m10n2	1545∼	12185	1467∼	11828	1545∼	12569	1510.25∼	12171
5m10n3	1652∼	12455	1598∼	12495	1707∼	13041	1673.75∼	12821.5
5m10n4	1638∼	12249	1598∼	12016	1648∼	12599	1637∼	12430
5m10n5	1510∼	11315	1427∼	10944	1510∼	11367	1495∼	11234.5
10m10n1	2117∼	18731	2047∼	17595	2137∼	18845	2122	18686.25
10m10n2	2112∼	18378	2057	18169	2152∼	18518	2107∼	18078
10m10n3	1950∼	17166	1903∼	17083	1955∼	17782	1938.5∼	17395.75
10m10n4	1970	17369	1935∼	16793∼	1975	17369	1970.87	17337.87
10m10n5	2047∼	17438	1981∼	17198∼	2015∼	17701	1989.5∼	17171.5∼
10m15n1	2399	30658	2356∼	30551	2435∼	31214	2412	30599.5
10m15n2	2600	33515	2540∼	32898	2600	33932+	2616.5	33875.25+
10m15n3	2503	32425+	2416∼	31585	2547	32806	2464.62∼	31681.12+
10m15n4	2492	31846	2405+	30998	2511	31979	2459.5	31710.25
10m15n5	2538	32693	2471+	32136	2552∼	33608+	2513.5	32637.5

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

As for OpH2, for each rule, the CPU time for constructing a schedule and calculating C_{max} and $\sum C_i$ values are ≤ 0.02 seconds.

Table 5.12 shows that, as for OpH2, the best rule for C_{max} is Rule (1); whereas the best one for $\sum C_i$ is Rule (2); these rules allow to obtain almost the best solutions. The worst rule for C_{max} is Rule (2) or (4) (resp. (2)) depending on the operations preemptive character and fixed (resp. flexible) unavailability periods; whereas the worst one for $\sum C_i$ is Rule (6). As for OpH2, the same conclusions are observed for Rules (3) and (4).

In general the same conclusions, as for job based heuristics, are observed for MOpH1 and MOpH2 concerning the dominance between the non-preemptive, resumable, non-resumable and semi-resumable problems (see Tables 5.13, 5.14, 5.15, and 5.16).

Table 5.7: Test results for flexible unavailability periods with OpH1 heuristic and 1000 iterations.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	825*	3576∼	825*	3576∼	825*	3576∼	825*	3576∼
5m5n2	1076*	4476∼	1076∼	4476∼	1076*	4476∼	1076∼	4476∼
5m5n3	1034*	4088∼	1034∼	4088∼	1034*	4088∼	1034∼	4088∼
5m5n4	1127∼	4332∼	1127∼	4332∼	1127∼	4332∼	1127∼	4332∼
5m5n5	1182*	4632∼	1182∼	4632∼	1182*	4632∼	1182∼	4632∼
5m10n1	1437	11088	1437∼	11088	1437∼	13059.55	1437∼	11088
5m10n2	1481∼	12026	1475∼	12026	1490∼	13615.21	1475∼	12026
5m10n3	1632∼	12667	1632∼	12667	1632∼	12667	1632∼	12667
5m10n4	1578∼	12108	1578∼	12108	1578∼	12108	1578∼	12108
5m10n5	1475∼	11411	1475∼	11411	1475∼	11411	1475∼	11411
10m10n1	2054∼	18207	2054∼	18207	2054∼	18207	2054∼	18207
10m10n2	2087	17954	2087∼	17954	2087	17954	2087	17954
10m10n3	1930∼	16763	1930	16763∼	1930∼	16763∼	1930∼	16763∼
10m10n4	1959	16616∼	1959	16616	1959	16616∼	1959	16616∼
10m10n5	1946∼	16949	1946∼	16949	1946∼	16949∼	1946∼	16949
10m15n1	2359	30384	2359	30384	2359∼	30384∼	2359∼	30384∼
10m15n2	2570∼	32598	2570∼	32598	2570∼	32598	2570∼	32598
10m15n3	2460	31005+	2460+	31005	2460∼	31600	2460+	31510
10m15n4	2412	30851	2412+	30851	2412	30851∼	2412∼	30851
10m15n5	2488	32343	2488∼	32343	2488	32343	2488∼	32343

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.8: Test results for fixed unavailability periods with OpH2 heuristic and the best solutions through all the rules.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1∼	895*	4094	895∼	3959	895*	4094	895∼	4004.5
5m5n2∼	1096*	4823∼	997*	4482∼	1096*	4706∼	1047.25∼	4604.5∼
5m5n3∼	1122∼	4735	1090∼	4570	1122∼	4735	1104.75∼	4640
5m5n4∼	1237∼	4890∼	1177∼	4136*	1237∼	4890∼	1198.5∼	4310∼
5m5n5∼	1290∼	5229	1168∼	4986	1280∼	5064∼	1272∼	5265
5m10n1∼	1560	11312	1533	10600∼	1560	11312	1560	11174.5
5m10n2∼	1495∼	12897	1460∼	12296	1495∼	12682	1496.5∼	12246
5m10n3∼	1877	12639	1740∼	12417	1877	12639	1839.5	12658.5
5m10n4∼	1767	11839	1617∼	11563∼	1738	11839	1661∼	12094
5m10n5∼	1615	11473	1486∼	10870	1615	11473	1548.25∼	11260.5
10m10n1∼	2121∼	18412∼	2021∼	18529	2178∼	18412∼	2084.5∼	18593.5
10m10n2∼	2214	19418	2218	18373	2202	19585	2212	19325
10m10n3∼	2062∼	18538	1943∼	17200	2102	18581	2062	18300
10m10n4∼	1990	18004	1898∼	16577∼	2040	18138	1959.12∼	17774.75
10m10n5∼	2115∼	17163∼	1977∼	16648∼	2115∼	17163∼	2100.5∼	17190∼
10m15n1∼	2420	29663∼	2377∼	28922∼	2437∼	29663∼	2501.75	29446.37∼
10m15n2∼	2585	33848	2466∼	33268	2586	33848+	2503.75∼	33122+
10m15n3∼	2621	31308+	2511∼	30690	2631	31308	2521	30666.5+
10m15n4∼	2452∼	30935	2441+	29943∼	2494	30935	2536.37	31041.5
10m15n5∼	2445∼	30911∼	2406+	30212∼	2445∼	30911	2488∼	31149.12∼

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.9: Test results for flexible unavailability periods with OpH2 heuristic and the best solutions through all the rules.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1∼	825*	3668∼	825*	3668∼	825*	3668∼	825*	3668∼
5m5n2∼	1076*	4747∼	1076∼	4747	1076*	4747∼	1076∼	4747
5m5n3∼	1042∼	4370∼	1042∼	4370	1042∼	4370∼	1042∼	4370
5m5n4∼	1217∼	4335∼	1217∼	4335∼	1217∼	4335∼	1217∼	4335∼
5m5n5∼	1211∼	4927∼	1211∼	4928	1211∼	4928∼	1211∼	4928∼
5m10n1∼	1540	10992	1540	10992	1540	10992	1540	10992
5m10n2∼	1525∼	12624	1525∼	12624	1525∼	12624	1525∼	12624
5m10n3∼	1740∼	12025∼	1740	12025	1740∼	12025	1740∼	12025∼
5m10n4∼	1628∼	11803	1628∼	11803	1628∼	11803	1628∼	11803
5m10n5∼	1595	12082	1595	12082	1595	12082	1595	12082
10m10n1∼	2011∼	17150∼	2011∼	17150∼	2011∼	17150∼	2011∼	17150∼
10m10n2∼	2147	18189	2147	18189	2147	18189	2147	18189
10m10n3∼	2042	18182	2042	18182	2042	18182	2042	18182
10m10n4∼	1911	16239∼	1911	16239∼	1911	16239∼	1911	16239∼
10m10n5∼	2005∼	16566∼	2005∼	16566∼	2005∼	16566∼	2005∼	16566∼
10m15n1∼	2338	30120	2338	30120	2338∼	30120+	2338∼	30120∼
10m15n2∼	2566∼	33794	2566∼	33794	2566∼	33794	2566∼	33794
10m15n3∼	2515	30318+	2515+	30318	2515∼	30318∼	2515	30318
10m15n4∼	2474	30082	2474+	30082∼	2474	30082∼	2474+	30082
10m15n5∼	2469	30419∼	2469∼	30419∼	2469	30419∼	2469	30419∼

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.10: Test results for modification of initial position of unavailability periods in case of resumable operations and flexible unavailability periods with OpH2 heuristic and through all the rules.

Problem	No modification of initial position of unavailability period		Unavailability period placed at the end of its time window		Unavailability period placed at the beginning of its time window		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	**825***	3668∼	**825***	3668∼	910∼	**3650**∼	825*	3356*
5m5n2	1076∼	4747	1076∼	4687	**977***	**4320**∼	977*	4174*
5m5n3	**1042**∼	4370	1100∼	4370	**1042**∼	**4296**∼	1020*	3892*
5m5n4	1217∼	4335∼	1217∼	4335∼	**1157**∼	**4136***	1108*	4136*
5m5n5	1211∼	4928	1211∼	4928	**1160**∼	**4790**∼	1108*	4417*
5m10n1	1540	10992	1540	10992	**1533**	**10521**	1316	9484
5m10n2	1525∼	12624	1525∼	12624	**1460**∼	**11951**	1400	10364
5m10n3	**1740**	**12025**	**1740**	**12025**	**1740**	12593	1563	10710
5m10n4	1628∼	11803	1628∼	11803	**1617**∼	**11090**∼	1492	10439
5m10n5	1595	12082	1595	12082	**1486**∼	**11048**	1372	9101
10m10n1	**2011**∼	**17150**∼	**2011**∼	**17150**∼	**2011**∼	17606∼	1912	16397
10m10n2	2147	18189	2147	18189	**2078**∼	**18168**	1915	15190
10m10n3	2042	18182	2042	18182	**1902**∼	**17176**	1726	15261
10m10n4	**1911**	16239∼	**1911**	16239∼	1927	**16193**∼	1701	14876
10m10n5	2005∼	**16566**∼	2005∼	**16566**∼	**1977**∼	16707∼	1839	15273
10m15n1	2338	30120	2338	30120	**2377**	**28343**∼	2087	26762
10m15n2	2566∼	33794	2566∼	33794	**2460**∼	**33116**	2450	28685
10m15n3	2515+	30318	2515+	30318	**2341**+	**30194**	4202	27221
10m15n4	2474+	**30082**∼	2515+	**30082**∼	**2441**+	30695	4941	27264
10m15n5	2469∼	30419∼	2515∼	30419∼	**2396**∼	**30360**∼	2292	28000

bold: best solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.11: Test results for resumable operations and flexible unavailability periods for orders ABC and CAB in OIp procedure with OpH2 heuristic and through all the rules.

Problem	OIp procedure				disjunctive MIP	
	order ABC		order CAB			
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	**825***	**3668∼**	845∼	3768	825*	3356*
5m5n2	**1076∼**	**4747**	1096	4777	977*	4174*
5m5n3	**1042∼**	**4370**	1062∼	4667	1020*	3892*
5m5n4	**1217∼**	**4335∼**	1237	4435∼	1108*	4136*
5m5n5	**1211∼**	**4928**	1280	5229	1108*	4417*
5m10n1	**1540**	**10992**	1560	11312	1316	9484
5m10n2	**1525∼**	**12624**	1545∼	13058	1400	10364
5m10n3	**1740**	**12025**	1745	12639	1563	10710
5m10n4	**1628∼**	**11803**	1648∼	11937	1492	10439
5m10n5	**1595**	**12082**	1615	12476	1372	9101
10m10n1	**2011∼**	**17150∼**	2031∼	17333∼	1912	16397
10m10n2	**2147**	**18189**	2167	18578	1915	15190
10m10n3	2042	**18182**	**2005**	18321	1726	15261
10m10n4	**1911**	**16239∼**	1990	16377	1701	14876
10m10n5	**2005∼**	**16566∼**	2025∼	16988	1839	15273
10m15n1	2338	30120	**2257∼**	**28439∼**	2087	26762
10m15n2	**2566∼**	33794	2585∼	**33160**	2450	28685
10m15n3	**2515+**	**30318**	2535+	30436	4202	27221
10m15n4	**2474+**	**30082∼**	2494+	30668	4941	27264
10m15n5	**2469∼**	**30419∼**	2489∼	30741∼	2292	28000

bold: best solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.12: Rules rank for MOpH2 and all availability models.

Rank	C_{max}		$\sum C_i$	
	Fixed unavailability periods	Flexible unavailability periods	Fixed unavailability periods	Flexible unavailability periods
1	1	2	1	2
2	5 or 6	4	4	4
3	5 or 6	1	6	1
4	2 or 4	3	5	3
5	3	5	3	5
6	2 or 4	6	2	6

Table 5.13: Test results for fixed unavailability periods with MOpH1 heuristic and 1000 iterations.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	895*	3891∼	845*	3738∼	895*	3891∼	866.62*	3798.5∼
5m5n2	1112∼	4706∼	1042∼	4385*	1112∼	4706∼	1077∼	4604.5∼
5m5n3	1070*	4411∼	1026∼	4082∼	1070*	4411∼	1049∼	4281.5∼
5m5n4	1205∼	4630∼	1147∼	4136*	1205∼	4630∼	1147∼	4307*
5m5n5	1202*	4683*	1160∼	4628∼	1280∼	4868∼	1198.75∼	4794∼
5m10n1	1420∼	10831∼	1350∼	10702	1460∼	11392	1445.87∼	10980.75
5m10n2	1525∼	12252	1434∼	11673	1525∼	12402	1495∼	11784.5
5m10n3	1682∼	12887	1598∼	12495	1682∼	13109	1665.5∼	12821.5
5m10n4	1617∼	12249	1560∼	12112	1638∼	12534	1638∼	12286.5
5m10n5	1495∼	11391	1425∼	10765	1510∼	11534	1486.5∼	11484
10m10n1	2101∼	18503∼	2072∼	18005∼	2142∼	18649∼	2082∼	18296∼
10m10n2	2119	17724	2057	17594	2112∼	17724∼	2098.87∼	17738.5
10m10n3	1992∼	17260	1918∼	16861∼	1992∼	17286∼	1921.75∼	16870.75∼
10m10n4	2004	17319	1897∼	16799∼	2004	17486	1970	17259
10m10n5	1985∼	17027∼	1972∼	16771∼	2023∼	17316∼	1986.25∼	17170.12∼
10m15n1	2443	31199	2348∼	29780∼	2405∼	31503	2327.37∼	30738.25
10m15n2	2572	32458	2576∼	32668	2572	33674+	2564.25∼	33321.25+
10m15n3	2467∼	32030+	2428∼	31142	2522	32030	2484.5∼	31498.37+
10m15n4	2454∼	31256	2363+	30352∼	2492	31561	2450.5	31364.81
10m15n5	2485∼	32479	2488+	31366	2548∼	32938	2499.75∼	32280.5

bold: best solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.14: Test results for flexible unavailability periods with MOpH1 heuristic and 1000 iterations.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	825*	3668∼	825*	3668∼	825*	3668∼	825*	3668∼
5m5n2	1076*	4537∼	1076∼	4537∼	1076*	4537∼	1076∼	4537∼
5m5n3	1034*	4177∼	1034∼	4177∼	1034*	4177∼	1034∼	4177∼
5m5n4	1127∼	4332∼	1127∼	4332∼	1127∼	4332∼	1127∼	5186.93
5m5n5	1182*	4626∼	1182∼	4626∼	1182*	4626∼	1182∼	5299.91
5m10n1	1430∼	10917	1397∼	10917	1430∼	10917	1430∼	10917
5m10n2	1469∼	11898	1469∼	11792	1469∼	11795	1469∼	11792
5m10n3	1622∼	12385	1622∼	12385	1622∼	12385	1622∼	12385
5m10n4	1578∼	12035	1578∼	12035	1578∼	12035	1578∼	12035
5m10n5	1475∼	11162	1475∼	11095	1475∼	11202	1475∼	11148.5
10m10n1	2057	17678∼	2057∼	17678∼	2057∼	17688∼	2057∼	17678∼
10m10n2	2097	17230	2097∼	17230	2097	17230	2097	17230
10m10n3	1901∼	16314∼	1901∼	16314∼	1901∼	16314∼	1901∼	16314∼
10m10n4	1920	16163∼	1920	16163∼	1920	16163∼	1920	16163∼
10m10n5	1955∼	16454∼	1955∼	16454∼	1955∼	16454∼	1955∼	16454∼
10m15n1	2360	30362	2360	30469	2360∼	30469+	2360∼	30362∼
10m15n2	2540∼	32510	2540∼	32510	2540∼	32510	2540∼	32510
10m15n3	2403∼	30848+	2403+	30848	2413∼	30848	2403∼	30848
10m15n4	2327∼	30857	2327+	30857	2327∼	30857∼	2327+	30857
10m15n5	2476	31651	2476∼	31651	2476	31651	2463	31651

bold: best solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.15: Test results for fixed unavailability periods with MOpH2 heuristic and through all the rules.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	895*	3947∼	845*	3738∼	895*	3947∼	895∼	3919,75∼
5m5n2	1112∼	4706∼	1080∼	4482∼	1112∼	4706∼	1096∼	4604,5∼
5m5n3	1122∼	4779	1110∼	4610	1122∼	4689	1116∼	4640
5m5n4	1255∼	4930∼	1268	4328∼	1375	4930∼	1338	4643,5∼
5m5n5	1280∼	5064∼	1168∼	4964	1280∼	5064∼	1239,75∼	4900,25∼
5m10n1	1560	11876	1560	10620	1560	11876	1560	11012,5
5m10n2	1635	12929	1460∼	11806	1605∼	13114	1535,25∼	12325,5
5m10n3	1884	12782	1770	12352	1884	12782	1849,5	12624
5m10n4	1638∼	11839	1621∼	11444∼	1738	11839	1638∼	12059,5
5m10n5	1625	11457	1486∼	10868	1625	11457	1548,25∼	11237,5
10m10n1	2141∼	19697	2151	18888	2228	19835	2117,75	18279∼
10m10n2	2296	17087∼	2132	16692∼	2296∼	17087∼	2262	16784,5∼
10m10n3	2072	18165	2000∼	17548	2211	18165	1992∼	17510
10m10n4	2110	17638	1978	16799∼	2095	17931	2009	17418,25
10m10n5	2055∼	16731∼	1977∼	17249∼	2055∼	16731∼	2115∼	17508,5
10m15n1	2420	29325∼	2377∼	28527∼	2500	29325∼	2405,25	29794,37∼
10m15n2	2533∼	34293	2550∼	33971	2702	33866+	2572,25∼	34614,87+
10m15n3	2551	31147+	2493∼	30850	2581	31147∼	2548,62	31139,31+
10m15n4	2494	32854	2441+	30610	2542	31738	2536,37	31492,25
10m15n5	2515∼	31267∼	2406+	31546	2515∼	31267	2489∼	32532,25

bold: best solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.16: Test results for flexible unavailability periods with MOpH2 heuristic and through all the rules.

Problem	Non-preemptive operations		Resumable operations		Non-resumable operations		Semi-resumable operations	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min
5m5n1	825*	3668∼	825*	3668∼	825*	3668∼	825*	3668∼
5m5n2	1076*	4605∼	1076∼	4605∼	1076*	4605∼	1076∼	4605∼
5m5n3	1070∼	4579	1070∼	4579	1070∼	4579	1070∼	4579
5m5n4	1235	4335∼	1235	4335∼	1235	4335∼	1235	4335∼
5m5n5	1211∼	4975∼	1211∼	4975	1211∼	4975∼	1211∼	4975∼
5m10n1	1540	11155	1540	11155	1540	11155	1540	11155
5m10n2	1475∼	11917	1475∼	11917	1475∼	11917	1475∼	11917
5m10n3	1794	12481	1794	12481	1794	12481	1794∼	12481
5m10n4	1618	11994∼	1618	11994∼	1618	11994∼	1618	11994∼
5m10n5	1577	11886	1577	11886	1577	11886	1577	11886
10m10n1	2156	18316	2156	18316	2156	18316	2156	18316
10m10n2	2202	16433∼	2202	16433∼	2202	16433∼	2202	16433∼
10m10n3	1972	17557	1972	17557	1972	17557	1972∼	17557
10m10n4	2070	16404∼	2070	16404∼	2070	16404∼	2070	16404∼
10m10n5	1995∼	16683∼	1995∼	16683∼	1995∼	16683∼	1995∼	16683∼
10m15n1	2400	29530∼	2400∼	29530	2400+	29530	2400	29530
10m15n2	2493∼	33350	2493∼	33350	2493∼	33350	2493∼	33350
10m15n3	2539	30338+	2539+	30338	2539∼	30338∼	2539+	30338
10m15n4	2474	30656	2474+	30656	2474	30656+	2474+	30656
10m15n5	2469	31879	2469∼	31879	2469	31879	2469	31879

bold: best solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

5.1.5.4 Comparison between all the heuristics

Tables 5.17 and 5.18 gather the test results for respectively non-preemptive operations and fixed unavailability periods, and resumable operation and flexible unavailability periods for all the heuristics. The results for JpH, OpH1 and MOpH1 are associated to the best ones through the different numbers if iterations 100, 1000, 10000 and 100000. For OpH2 and MOpH2, the results are associated to the best ones through all the rules from 1 to 6.

Table 5.17: Comparing the heuristics in case of non-preemptive operations and fixed unavailability periods.

Problem	JpH		OpH1		OpH2		MOpH1		MOpH2		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	*961*∼	3748∼	**895***	**3652***	**895***	*4094*	**895***	3874∼	**895***	3947∼	895*	3652*
5m5n2	*1112*∼	**4669***	**1096***	**4669***	**1096***	*4823*∼	*1112*∼	4706∼	*1112*∼	4706	1096*	4669*
5m5n3	*1157*∼	**4008***	**1070***	**4008***	1122∼	4735	**1070***	4130∼	1122∼	*4779*	1070*	4008*
5m5n4	**1147***	**4584***	**1147***	**4584***	1237∼	4890∼	1205∼	4630∼	*1255*	*4930*	1147*	4584*
5m5n5	**1202***	**4663***	**1202***	**4663***	*1290*∼	*5229*∼	**1202***	**4663***	1280∼	5064∼	1202*	4663*
5m10n1	1471∼	**10304**∼	**1420**∼	10759∼	*1560*	11312	**1420**∼	10574∼	*1560*	*11876*	1361*	9850*
5m10n2	1521∼	**11036**∼	**1455**∼	11775	1495∼	12897	1485∼	11817	*1635*	*12929*	1455	10637*
5m10n3	1722∼	**11594**∼	**1607**∼	12234∼	1877	12639	1632∼	12057∼	*1884*	*12782*	1607	11072
5m10n4	1619∼	**11353**∼	**1557**∼	11813	*1767*	*11839*	1577∼	11683∼	1638∼	*11839*	1537*	10630*
5m10n5	1515∼	**10179**∼	**1425**∼	10410∼	1615	11473	1435∼	10898	*1625*	*11457*	1415	9706*
10m10n1	*2226*	**17308**∼	**2030**∼	17935∼	2121∼	18412∼	2047∼	17896∼	2141∼	*19697*	1948	16878
10m10n2	2202	**16539**∼	**2022**∼	17342∼	2214	*19418*	**2022**∼	17247∼	*2296*	17087∼	1917*	15777*
10m10n3	*2098*	16238∼	**1912**∼	16641∼	2062∼	*18538*	**1912**∼	**16235**∼	2072	18165	1875	15352*
10m10n4	1990	**16143**∼	**1860**∼	16331∼	1990	*18004*	1904∼	16571∼	*2110*	17638	1772*	15374*
10m10n5	2105∼	**16132**∼	**1975**∼	16795∼	*2115*	*17163*	**1975**∼	16559∼	2055∼	16731∼	1975	15625*
10m15n1	*2550*	**28661**∼	2320	*30071*	2420	29663∼	**2274**∼	29703∼	2420	29325	2089	27199
10m15n2	*2742*	**30743**∼	2490∼	32194∼	2585	33848	**2480**∼	32021∼	2533∼	*34293*	2316	29186
10m15n3	2513	**28814**+	2423∼	31213+	*2621*	*31308*+	**2401**∼	30698+	2551	31147+	2261	u
10m15n4	*2532*	**28436**∼	2316∼	30781	2452∼	30935	**2312**∼	30168∼	2494	*32854*	2254	27423
10m15n5	*2662*	**30114**∼	2462∼	*31787*	2445∼	30911∼	**2422**∼	31194∼	2515∼	31267∼	2282	28609

bold: best solution

*: optimal solution

u: unknown solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.18: Comparing the heuristics in case of resumable operations and flexible unavailability periods.

Problem	JpH		OpH1		OpH2		MOpH1		MOpH2		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	**825***	3597∼	**825***	**3552**∼	**825***	3668∼	**825***	*3668*∼	**825***	*3668*∼	825*	3356*
5m5n2	**1076**∼	4468∼	**1076**∼	4412∼	**1076**∼	*4747*	**1076**∼	4537∼	**1076**∼	4605∼	977*	4174*
5m5n3	1137	3968∼	**1034**∼	3968∼	1042∼	4479	**1034**∼	4012∼	*1070*∼	*4579*	1020*	3892*
5m5n4	**1108***	**4242**∼	**1108***	**4242**∼	1217∼	*4335*∼	**1108***	4332∼	*1235*	*4335*∼	1108*	4136*
5m5n5	1190∼	4646∼	**1182**∼	4565∼	*1211*∼	4928	**1182**∼	4626∼	*1211*∼	*4975*	1108*	4417*
5m10n1	1440∼	**10119**∼	**1360**∼	10448∼	*1540*	10992	**1360**∼	10389∼	*1540*	*11155*	1316	9484
5m10n2	1464∼	**10778**∼	**1404**∼	11388∼	*1525*	*12624*	1409∼	11398	1475∼	11917	1400	10364
5m10n3	1702∼	**11307**∼	1587∼	11805∼	1740	12025	**1578**∼	11811∼	*1794*	*12481*	1563	10710
5m10n4	1618∼	**10975**∼	**1518**∼	11327∼	*1628*	11803	**1518**∼	11519∼	1618∼	*11994*	1492	10439
5m10n5	1486∼	**9860**∼	**1405**∼	10533	*1605*	*12082*	**1405**∼	10081	1577	11886	1372	9101
10m10n1	*2202*	17348∼	**1991**∼	17614∼	2011∼	17150∼	**1911**∼	**17269**∼	2156	*18316*	1912	16397
10m10n2	2136	16448∼	**1982**∼	16855	2147	*18189*	1992∼	16766∼	*2202*	**16433**∼	1915	15190
10m10n3	2031	**15829**∼	1836∼	16205∼	*2042*	*18182*	**1832**∼	15962∼	1972	17557	1726	15261
10m10n4	1970	**15772**∼	**1847**∼	16033∼	1911	16239∼	1853∼	15975∼	*2070*	*16404*	1701	14876
10m10n5	*2032*∼	**15697**∼	**1913**∼	16153∼	2005∼	16566∼	**1913**∼	16041∼	1995∼	*16683*	1839	15273
10m15n1	*2549*	**28192**∼	2282∼	29521∼	2338	*30120*	**2259**∼	28903∼	2400	29530∼	2087	26762
10m15n2	*2596*	**30074**∼	**2440**∼	31752	2566∼	*33794*	2458∼	31641∼	2493∼	33350	2450	28685
10m15n3	2517+	**28619**∼	**2277**+	30292	2515+	30318	2342+	29945	*2539*+	*30338*	4202	27221
10m15n4	*2552*+	**28379**∼	2342+	30095∼	2474+	30082∼	**2302**+	29439∼	2474+	*30656*	4941	27264
10m15n5	*2552*	**29486**∼	2376∼	31389	2469∼	30419∼	**2364**∼	30541∼	2469∼	*31879*	2292	28000

bold: best solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table 5.19: Heuristics rank.

Rank	C_{max}		$\sum C_i$	
	Non-preemptive operations	Resumable operations	Non-preemptive operations	Resumable operations
	Fixed unavailability periods	Flexible unavailability periods	Fixed unavailability periods	Flexible unavailability periods
1	OpH1	MOpH1	JpH	JpH
2	MOpH1	OpH1	OpH1	OpH1
3	OpH2	JpH or OpH2	MOpH1	MOpH1
4	MOpH2	JpH or OpH2	OpH2	OpH2
5	JpH	MOpH2	MOpH2	MOpH2

The ranking of Table 5.19 is deduced from Tables 5.17 and 5.18. Then the best heuristics for C_{max} are OpH1 and MOpH1; OpH1 is dominant in case of non-preemptive operations and fixed unavailability periods; and MOpH1 is slightly dominant in case of resumable operations and flexible unavailability periods. The worst heuristic for C_{max} is JpH in case of non-preemptive and fixed unavailability periods; while JpH is largely dominant for $\sum C_i$. For this criterion, dominances of OpH1 and MOpH1 are quite equal with a slight dominance of OpH1. The worst results are largely associated to heuristics OpH2 and MOpH2; however MOpH2 is slightly worse than OpH2. These results prove that it is better to use randomness in choosing initial sequences of jobs and operations than to use the rules of OpH2 and MOpH2.

Table 5.20: Comparison of heuristics JpH and JpH-R2 with 100000 iterations.

Problem	Non-preemptive operations and fixed unavailability periods						Resumable operations and flexible unavailability periods					
	JpH		JpH-R2		MIP		JpH		JpH-R2		MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	961	**3748**	**895***	4247	895*	3652*	**825***	**3597**	871	3779	825*	3356*
5m5n2	1112	**4669***	**1096***	4927	1096*	4669*	**1076**	**4468**	**1076**	4607	977*	4174*
5m5n3	1157	**4008***	**1070***	4675	1070*	4008*	1137	**3968**	**1034**	4527	1020*	3892*
5m5n4	**1147***	**4584***	1327	5515	1147*	4584*	**1108***	**4242**	1207	4756	1108*	4136*
5m5n5	**1202***	**4663***	1280	5064	1202*	4663*	**1190**	**4646**	1211	4938	1108*	4417*
5m10n1	1471	**10304**	**1420**	11330	1361*	9850*	1440	**10119**	**1340**	11081	1316	9484
5m10n2	**1521**	**11036**	1545	13042	1455	10637*	**1464**	**10778**	1525	12259	1400	10364
5m10n3	1722	**11594**	**1637**	13483	1607	11072	1702∼	**11315**	**1617**	13471	1563	10710
5m10n4	**1619**	**11353**	1688	13742	1537*	10630*	**1618**	**10975**	1638	13256	1492	10439
5m10n5	**1515**	**10179**	1615	12652	1415	9706*	**1486**	**9860**	1595	11994	1372	9101
10m10n1	2231	**17341**	**2097**	19122	1948	16878	2202	**17348**	**1962**	17905	1912	16397
10m10n2	2202	**16539**	**2072**	18222	1917*	15777*	2136	**16448**	**2052**	17995	1915	15190
10m10n3	2098	**16238**	**1955**	18280	1875	15352*	2031	**15829**	**1855**	16979	1726	15261
10m10n4	1990	**16143**	**1900**	17483	1772*	15374*	1970	**15772**	**1840**	16893	1701	14876
10m10n5	2105	**16182**	**1975**	17288	1975	15625*	2032	**15697**	**1913**	16682	1839	15273
10m15n1	2594	**28661**	**2319**	31090	2089	27199	2549	**28192**	**2244**	30839	2087	26762
10m15n2	2742	**30743**	**2440**	33072	2316	29186	2596	**30074**	**2420**	32803	2450	28685
10m15n3	2513	**28814**	**2421**	32497	2261	u	2517	**28619**	**2391**	31759	4202	27221
10m15n4	2532	**28436**	**2284**	31506	2254	27423	2552	**28379**	**2271**	30633	4941	27264
10m15n5	2662	**30114**	**2409**	32474	2282	28609	2552	**29486**	**2328**	31988	2292	28000

bold: best solution

*: optimal solution

u: unknown solution

From 5.20, we deduce that the value of C_{max} are improved by JpH-R2 heuristic for many instances specially the big ones comparatively to JpH heuristic and almost values of the two heuristics are close. However, the results of JpH-R2 for $\sum C_i$ are worst.

Table 5.21: Comparing results for JpH-R2 heuristic and the best values over all the other heuristics.

Problem	Non-preemptive operations and fixed unavailability periods						Resumable operations and flexible unavailability periods					
	Best values		Jph-R2		MIP		Best values		JpH-R2		MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	**895***	**3652***	**895***	4247	895*	3652*	**825***	**3552**	871	3779	825*	3356*
5m5n2	**1096***	**4669***	**1096***	4927	1096*	4669*	**1076**	**4412**	**1076**	4607	977*	4174*
5m5n3	**1070***	**4008***	**1070***	4675	1070*	4008*	**1034**	**3968**	**1034**	4527	1020*	3892*
5m5n4	**1147***	**4584***	1327	5515	1147*	4584*	**1108***	**4242**	1207	4756	1108*	4136*
5m5n5	**1202***	**4663***	1280	5064	1202*	4663*	**1182**	**4565**	1211	4938	1108*	4417*
5m10n1	**1420**	**10304**	**1420**	11330	1361*	9850*	1360	**10119**	**1340**	11081	1316	9484
5m10n2	**1455**	**11036**	1545	13042	1455	10637*	**1404**	**10778**	1525	12259	1400	10364
5m10n3	**1607**	**11594**	1637	13483	1607	11072	**1578**	**11307**	1617	13471	1563	10710
5m10n4	**1557**	**11353**	1688	13742	1537*	10630*	**1518**	**10975**	1638	13256	1492	10439
5m10n5	**1425**	**10179**	1615	12652	1415	9706*	**1405**	**9860**	1595	11994	1372	9101
10m10n1	**2030**	**17308**	2097	19122	1948	16878	**1911**	**17348**	1962	17905	1912	16397
10m10n2	**2022**	**16539**	2072	18222	1917*	15777*	**1982**	**16448**	2052	17995	1915	15190
10m10n3	**1912**	**16235**	1955	18280	1875	15352*	**1832**	**15829**	1855	16979	1726	15261
10m10n4	**1860**	**16143**	1900	17483	1772*	15374*	1853	**15772**	**1840**	16893	1701	14876
10m10n5	**1975**	**16132**	**1975**	17288	1975	15625*	**1913**	**15697**	**1913**	16682	1839	15273
10m15n1	**2274**	**28661**	2319	31090	2089	27199	2259	**28192**	**2244**	30839	2087	26762
10m15n2	2480	**30743**	**2440**	33072	2316	29186	2440	**30074**	**2420**	32803	2450	28685
10m15n3	**2401**	**28814**	2421	32497	2261	u	**2277**	**28619**	2391	31759	4202	27221
10m15n4	2312	**28436**	**2284**	31506	2254	27423	2302	**28379**	**2271**	30633	4941	27264
10m15n5	2422	**30114**	**2409**	32474	2282	28609	2364	**29486**	**2328**	31988	2292	28000

bold: best solution

italic: worst solution

*: optimal solution

u: unknown solution

$\sim$: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

The comparison between the values of JpH-R2 and the best values over all the other heuristics (see Table 5.21) shows that, for C_{max} criterion, the values for some benchmarks (specially big ones) are slightly better with JpH-R2. However, the values of $\sum C_i$ are worst.

5.2 Using construction heuristics in improving methods

Remark:

As the construction heuristics presented previously are very fast, they can be all implemented in another method. The solution to the problem is the best one over the solutions of the heuristics.

5.2.1 Reoptimizing OpH1 (reOpH1)

The construction of a schedule is made one operation after the other. According to an initial priority sequence s for operations $(O_1, .., O_{no})$, where $no = \sum_{i=1}^{n} n_i$ is the total number of operations. Each time that an operation O from sequence s is sequenced on its associated machine by OIp procedure, all data structures, corresponding to the unavailability periods, the availability intervals and the starting and completion dates for all the sequenced operations (priority sequence $s_O = (1, .., O-1)$), are reinitialized. Operation O is fixed on its associated machine as an unavailability period; OpH1 heuristic is reused to schedule the priority sequence s_O.

Algorithm 9 describes the main steps of the heuristic.

For this method also the performance of the heuristic strongly depends on the operation priority order in the initial sequence.

Algorithm 9 Algorithm of reoptimizing Operation priority Heuristic 1 (reOpH1)

Begin
  Find an initial operation priority sequence s
    for each operation O in the sequence s
      Use the procedure OIp to insert operation O on its machine
      Reinitialize all data structures for unavailability periods,
      availability intervals and starting and completion dates for all
      the sequenced operation priority sequence $s_O = (1, .., O-1)$)
      Fix O on its associated machine as an unavailability period
      Use OpH1 to schedule the priority sequence s_O
    end for
end

5.2.2 Genetic Algorithm

We have previously underlined that heuristics JpH, OpH1 and MOpH1 depend strongly on an initial sequence. The test results show that higher randomness in the sequence, better are the solutions. The high number of runs (iterations) of these methods show that it is interesting to consider un wide solution space. All these motivates the design of an improving method such as the genetic algorithm. This method is interesting because it works with a population of sequences.

This approach is similar to the one developed by Aggoune [Agg02]. However, in addition to the introduction to the flexibility on the starting date of the unavailability periods, another chromosome coding than the job priority sequence is also used: the operation priority sequence.

The makespan evaluation is made by JpH (resp. OpH1 or MopH1) for the job priority sequence (resp. operation sequence).

Algorithm 10 describes the main steps of the genetic algorithm.

Algorithm 10 The genetic algorithm

Begin
    Define a coding and generate an initial population
    Evaluate each individual of the population
    while a stopping condition is not satisfied
        Select individuals for recombining
        Apply variation operators (crossover, mutation) on the selected individuals
        Evaluate the performance of the new individuals
        Replace individuals to get the new population
    end while
end

5.3 Conclusion

As approximation methods are appropriate alternative for exact methods for solving NP-hard combinatorial optimization problems, we have presented in this Chapter heuristics to tackle the job shop scheduling problem with resource availability constraints. Introducing flexibility on starting dates of unavailability periods and operations preemption was studied.

Construction heuristics we developed construct a schedule based on various decision strategies. The choice of these strategies are related to how operations and/or machines are prioritized, and how conflicts between operations and machine unavailability periods are managed. Difficulties are how to select an operation to be inserted on the corresponding machine, how to

select the availability period on the machine that can contain the operation, and how to deal with operations preemption and unavailability periods flexibility.

Two kinds of methods are suggested: job based and machine based heuristics. Three job based heuristics that prioritize the job operations were developed. In the first two heuristics, Job priority Heuristic (JpH) and Operation priority Heuristic 1 (OpH1), selection is based on the ordering of the jobs and the operations respectively. In the third heuristic OpH2, the operation to be inserted is selected according to a given rule. Two machine based heuristics were also proposed.

Finally, we have discussed the way these construction heuristics, that are building blocks, are used in improving methods to obtain better results for the studied problem. JpH, OpH1 and MOpH1 can be used as evaluating blocks in methods that use many solutions at the same time to provide better ones as it is the case for genetic algorithms. OpH1 and MOpH1 can be re-used in a method that reoptimizes the sub-sequence formed by the previous operations in the initial sequence, each time an operation is selected and inserted.

Chapter 6

Column Generation Approach

In this chapter, we propose a column generation approach to solve the classical job shop scheduling problems with fixed and flexible resource availability constraints. Various objective functions are used. A new integer programming formulation is proposed, where variables are associated to the selection of a schedule for a given job or of a schedule for a given resource unavailability period. Because the number of variables is huge, a column generation approach is developed to only select relevant schedules until convergence is obtained.

The chapter is organized as follows: In Section 6.1, we present the main idea of the column generation approach and discuss interests in using large-scale mixed integer programming problems and column generation models. Section 6.2 presents a column generation approach for the non-preemptive job shop problem without and with fixed resource availability periods; that is extended to consider preemptive problems. This approach is adapted in Section 6.3 to take into account flexible unavailability periods; obviously this approach solves as a special case the problem with fixed unavailability periods.

6.1 Why Column Generation?

In Barnhart *et al.* [BJNSV98], it is mentioned that the successful resolution of large-scale mixed integer programming problems (MIP) requires formulations whose linear programming (LP) relaxations give a good approximation of the convex hull of feasible solutions. It is also mentioned that, in column generation, sets of columns are left out of the LP relaxation because there are too many columns to handle efficiently and most of them will have their associated variables equal to zero in an optimal solution anyway.

Hence, the column generation approach is based on an (MIP) formulation inducing a huge number of variables; and is performed with the linear relaxation program of the MIP. Starting from an initial set of columns that corresponds to a feasible solution to the problem, columns are iteratively added to the reduced problem thanks to a pricing problem that constructs feasible schedules corresponding to improving columns. The reduced problem is reoptimized at each iteration. The process is stopped when no column is added. If the solution to the reduced

problem is integer, it is also the solution to the MIP; otherwise, the integrality property of the variables is introduced to the reduced problem that is then solved by a branch-and-bound procedure or a branch-and-price procedure.

In Lancia *et al.* [LRS07], a compact formulation of a model with an exponential number of constraints is defined as an equivalent formulation in which the exponentially many constraints are replaced by a polynomial number of new constraints (after introducing an exponential number of new variables).

In Barnhart *et al.* [BJNSV98], several reasons for considering formulations with a huge number of variables are listed:

- When a compact formulation of a MIP may have a weak LP relaxation, the relaxation can be tightened by a reformulation that involves a huge number of variables,
- A formulation with a huge number of variables can eliminate the symmetry that can exist in the structure of a compact formulation of a MIP. This symmetry causes a poor performance of branch-and-bound,
- Column generation provides a decomposition of the problem into a master problem and a pricing problem. This decomposition may have a natural interpretation in the contextual setting allowing for the incorporation of additional important and complex constraints,
- A formulation with a huge number of variables may be the only choice.

Section 6.2 develops the column generation approach for the non-preemptive job shop problem with fixed availability periods of machines, and an extension of the approach to the preemptive problem is presented. Section 6.3 studies the problem in case of flexible availability periods.

6.2 Non-preemptive job shop problem with fixed resource availability periods

A similar approach was proposed by Lancia *et al.* [LRS07] to solve the classical job shop problem. As the time required to reach optimality of the related model by column generation can be very large, they proposed a compact formulation. No indications are given on how the column generation approach is developed and implemented and what are the difficulties and properties.

Our approach is based on the integer programming formulation presented in Section 6.2.1. Section 6.2.2 describes the column generation approach; and the main steps are given in Section 6.2.3. The implementation and test results are discussed in Section 6.2.4. Finally, an extension of the approach to the preemptive problem is presented in Section 6.2.5.

6.2.1 An integer programming formulation

We want to develop a formulation of the problem that will allow us to use column generation. We denote by T the schedule length. A solution of the problem can be described by the n schedules of the jobs on the machines. $S(J_i)$ is the set of sequences of job J_i. It represents the (exponential) number of all the possible *feasible* schedules of J_i.

Illustration example of schedule sets

For illustration, let us consider the example of the problem given in Figure 6.1. The production system is composed of two machines M_1 and M_2 to process two jobs J_1 and J_2. The job routing of J_1 is M_1 (3) M_2 (2) and the one of J_2 is M_2 (1) M_1 (2). The schedule length we choose is $T = 8$ and we decompose the time in 8 periods.

As we assume that there no unavailability period on the machines, a feasible schedule of a job is a schedule that respects the job routing. Hence, sets of schedules $S(J_1)$ and $S(J_2)$ associated respectively to jobs J_1 and J_2 are composed respectively of 10 and 21 schedules that are feasible for their associated jobs. Each of these schedules is represented by a sequence of 1 and 0. Bit 1 means that the job is being processed at the associated period. If it is not the case, bit 0 is associated. Moreover, a number is assigned to each schedule; the 10 first schedules concern job J_1 whereas numbers from 11 to 31 are assigned to schedules of J_2. Let us consider schedule number 7, operation O_{11} starts at period 2 and finishes at period 4; and operation O_{12} starts and finishes respectively at periods 7 and 8. S is the set of all feasible schedules ($S = S(J_1) \cup S(J_1)$).

To determine a feasible solution to the problem, it is necessary to find a schedule from $S(J_1)$ and a schedule from $S(J_2)$, that are also feasible for the global problem as in addition to the respect of the job routings, they must also satisfy the resource constraints (non overlapping of two operations on the same machine). For instance, schedule 7 of J_1 and schedule 14 of J_2 are feasible for the global problem; that is not the case of schedules 7 and 17 as operations O_{21} and O_{22} overlap on machine M_1 during periods 3 and 4.

Note that the higher is the schedule length, the higher is the number of schedules for each job and inversely. For instance if we set $T = 6$, only schedules 1, 2, 5 remain feasible for J_1 and only schedules 11, 12, 13, 14, 17, 18, 19, 22, 23, 26 are still valid for J_2.

Notations

In order to introduce our formulation, we need the following parameters:

n: number of jobs,

m: number of machines,

S: the set of all schedules ($S = \bigcup_{i=1}^{n} S(J_i)$),

$J_i(s)$: the job associated to a given schedule s, i.e. $s \in S(J_i)$; for instance, in Figure 6.1, the schedule $s = 7$ is associated to job J_1 and schedule $s = 18$ to J_2 ,

$a_{tj}^{s} = 1$ if the j^{th} operation of job $J_i(s)$ is processed at period t on its associated machine

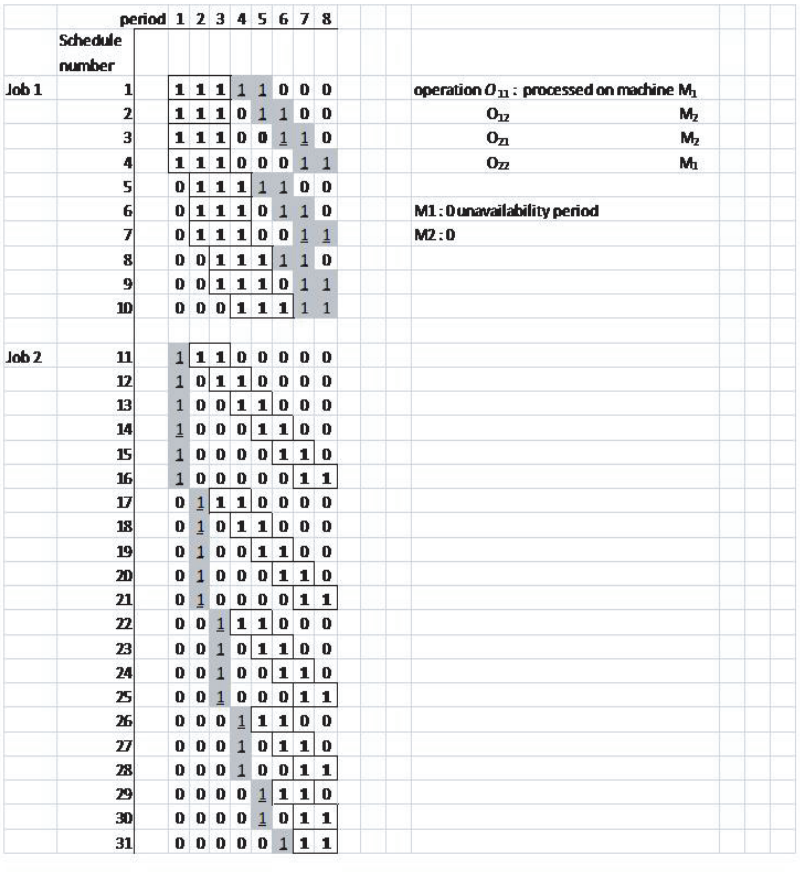

Figure 6.1: Example of schedule set S for a problem without machine unavailability periods.

and 0 otherwise,

C_s: cost associated to schedule s (for instance, the completion time of job $J_i(s)$),
n_i: number of operations of job J_i,
p_{ij}: processing time of operation O_{ij} of job J_i,
mr_{ij}: machine on which operation O_{ij} of job J_i is processed.

Remark

Although this formulation models the job shop with continuously available machines, it also models the problem with machine availability constraints. In addition to the properties that should satisfy any schedule in S without machine unavailability period, the operations should be sequenced such that no operation overlaps an unavailability period on its associated machine. Obviously the set of schedules when unavailability periods is included in the set of schedules without unavailability periods (See Figures 6.1 and 6.2). As mentioned before for

the example of Figure 6.1, in case of machines that are continuously available, the feasibility of a schedule associated to a job is conditioned by the its respect to the job routing; however in case of unavailability periods on the machines , refer to Figure 6.2, an additional condition is the processing of operations only during availability periods of the machines. For example, if we introduce an unavailability period on machine M_1 at periods 4 and 5, only schedules 1, 2, 3 and 4 remain feasible for J_1 and only schedules 11, 15, 16, 20, 21, 24, 25, 27, 28, 29, 30, 31 for J_2. And if we introduce also an unavailability period on machine M_2 at periods 4, schedules 3, 4 are no more feasible for J_1.

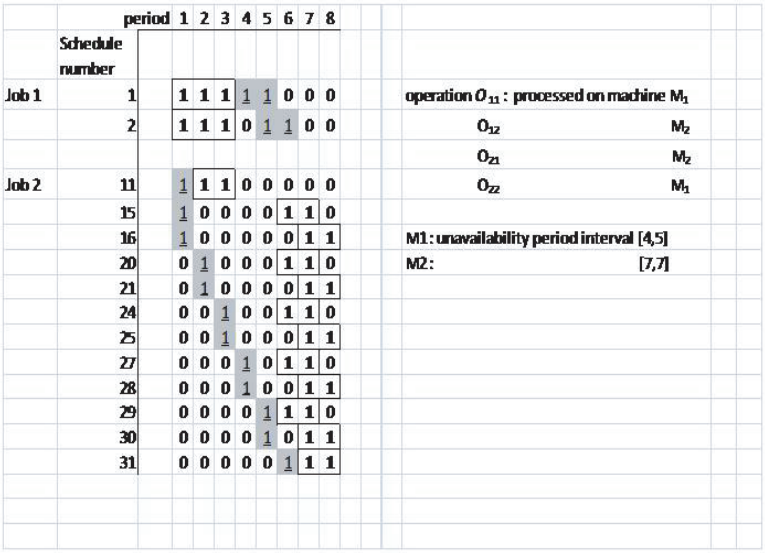

Figure 6.2: Example of schedule set S for the problem of Figure 6.1 with machine unavailability periods.

The model

The following variables are used in the model:

$x_s = 1$ if the s^{th} feasible schedule is used for job $J_i(s)$ and 0 otherwise.

The integer programming formulation, called (MP), is:

$$f^* = \min f = \sum_{s \in S} C_s x_s \tag{6.1}$$

$$\sum_{s \in S(J_i)} x_s = 1 \qquad i = 1, .., n \tag{6.2}$$

$$\sum_{s \in S} \sum_{j=1; m r_{J_i(s)j} = r}^{n_{J_i(s)}} a^s_{tj} x_s \leq 1 \qquad t = 1, .., T; r = 1, .., m \tag{6.3}$$

$$x_s \in \{0, 1\} \qquad \forall s \in S \tag{6.4}$$

Constraint [6.2] ensures that one and only one schedule is selected for each job. Constraint [6.3] guarantees that it is never used by more than one operation at each period. The number of variables is $|S|$, which can quickly become very large, even for small values of n and m. It has respectively n and $T \times m$ constraints associated to Constraints [6.2] and [6.3].

Note that the criterion can be very general as long as it only depends on each job: sum of the (weighted) completion times $\sum_s C_s x_s$ ($\sum_s w_{J_i(s)} C_s x_s$), sum of the (weighted) tardiness $\sum_s T_s x_s$ ($\sum_s w_{J_i(s)} T_s x_s$) and/or earliness $\sum_s E_s x_s$ ($\sum_i w_{J_i(s)} E_s x_s$),... Where $T_s = max(C_s - d_{J_i(s)}, 0)$ and $E_s = max(d_{J_i(s)} - C_s, 0)$. It can actually be non linear. As the criterion C_{max} depends on all the jobs, the column generation approach cannot be applied.

6.2.2 A column generation approach

Designing a column generation approach depends on the development of the following elements: the relaxation of the integer programming formulation seen in the previous section (Section 6.2.2.1), the dual problem associated to this relaxation (Section 6.2.2.2), the pricing problem defining the form of an improving column (Section 6.2.2.3) to add to the relaxed problem and the dynamic programming procedure to find the columns to add (Section 6.2.2.4).

6.2.2.1 Master Problem

Our initial goal is to solve the linear relaxation of (MP) i.e. $x_s \in [0, 1], \forall s \in S$. We suppose that the model is first solved for an initial set of schedules to which columns will be added. This is called *Reduced Master Problem* (RMP).

When applying column generation, it is preferable (see Barnhart *et al.* [BJNSV98]) to use a covering formulation than a partitioning formulation. Hence, in the sequel, we replace the equality sign (=) in Constraint [6.2] by larger than or equal to ($\geq$).

6.2.2.2 Dual Problem

Dual variables are obtained for every constraint of the Master Problem, denoted by γ_i for the n constraints [6.2] and δ_{tr} for the $T \times m$ constraints [6.3].

The dual of the linear relaxation of (MP) problem can be written:

$$d^* = \max d = \sum_{i=1}^{n} \gamma_j - \sum_{t=1}^{T} \sum_{r=1}^{m} \delta_{tr} \tag{6.5}$$

$$\gamma_{J_i(s)} - \sum_{t=1}^{T} \sum_{j=1}^{n_{J_i(s)}} a_{tj}^s \delta_{tmr_{ij}} \leq C_s \qquad \forall s \in S \tag{6.6}$$

$$\gamma_i \geq 0 \qquad i = 1, .., n \tag{6.7}$$

$$\delta_{tr} \geq 0 \qquad t = 1, .., T; r = 1, .., m \tag{6.8}$$

The dual variable γ_i represents the feasibility of job J_i, and δ_{tr} corresponds to the state of resource r at period t.

6.2.2.3 Pricing Problem

Adding a column (i.e. a feasible schedule) to the relaxed primal problem corresponds to adding a constraint [6.6] to the dual. Decreasing the objective value of the primal will lead to a decrease of the objective value of the dual problem. This can only be done if the new constraint [6.6] is not satisfied for the optimal values γ_i^* and δ_{tr}^* of the dual variables. Hence, we are searching for a schedule s such that:

$$\gamma_{J_i(s)}^* - \sum_{t=1}^{T} \sum_{j=1}^{n_{J_i(s)}} a_{tj}^s \delta_{tmr_{ij}}^* > C_s \tag{6.9}$$

The reasoning leading to [6.9] is based on the dual variables corresponding to all processing periods of the operations of job J_i.

The search needs to be done for every job J_i since the constraint differs from one job to another.

Let us denote by c_{tj}^s the boolean parameter which is equal to 1 if operation O_{ij} $(1 \leq j \leq n_i)$ of job J_i ends at period t. Consider from now that C_s corresponds to the completion time of job J_i in the schedule, i.e. the end of the last operation of job $J_i(s)$: $C_s = \sum_{t=1}^{T} t c_{t.n_{J_i(s)}}^s$.

Constraint [6.9] can be rewritten

$$\gamma_{J_i(s)}^* - \sum_{t=1}^{T} \sum_{j=1}^{n_{J_i(s)}} c_{tj}^s \Big(\sum_{l=t-p_{ij}+1}^{t} \delta_{lmr_{ij}}^* \Big) > \sum_{t=1}^{T} t c_{tn_{J_i(s)}}^s \tag{6.10}$$

The reasoning leading to [6.10] is based on the dual variables corresponding to processing period referring to the completion date of each operation of job J_i.

If operation o of job J_i ends at period t in a schedule s, then the contribution of the operation at period t is

$$w_{tj} = \sum_{l=t-p_{ij}+1}^{t} \delta_{lmr_{ij}}^*, \quad \text{if} \quad j < n_i$$

$$w_{tj} = t + \sum_{l=t-p_{ij}+1}^{t} \delta^*_{lmr_{ij}}, \quad \text{if} \quad j = n_i$$

Using the previous notation, [6.10] is equivalent to

$$\sum_{t=1}^{T} \sum_{j=1}^{n_{J_i(s)}} c^s_{tj} w_{tj} < \gamma^*_{J_i(s)} \tag{6.11}$$

For every job J_i, we are searching for a schedule s verifying [6.11]. Indeed, if there is no such schedule, then no column needs to be added.

Remarks

The contributions of the job operations are not only useful to construct a feasible schedule, but also to define the improving columns to add to (RMP). Obviously, there are some periods at which operation O_{ij} cannot finish. These periods must be discarded by setting to ∞ the associated contributions. This can be done to satisfy the job routing (this suggests the definition of a lower bound and an upper bound; that are unfortunately not tight since we deal with the job-shop scheduling problem); and to ensure non overlapping between an operation and an unavailability period.

This defines a pretreatment phase. Note that even if this phase is not performed, the discarded periods will be skipped when constructing the schedule by the dynamic programming algorithm.

1. *Job routing constraints*

   1.a. Operation O_{ij} cannot be completed before the completion of the previous operation in the job routing and its processing time. Hence, a lower bound for the completion period of O_{ij} is given by $B_l = \sum_{j'=1}^{j} p_{ij'}$. It is expressed by the sum of the durations of the first j operations. Then,

   $$w_{tj} = +\infty, \quad \forall t = 1, .., B_l - 1$$

   1.b. By analogy, operation O_{ij} cannot be completed after the starting time of the next operation in the job routing. Hence, an upper bound for the completion period of O_{ij}, is given by $B_u = T - (\sum_{j'=j+1}^{n_i} p_{ij'}) - 1$. It is deduced from the sum of the durations of the $n_i - j$ last operations. Hence,

   $$w_{tj} = +\infty, \quad \forall t = B_u + 1, .., T$$

2. *Constraints from machine unavailability periods.*

2.a. Operation O_{ij} cannot be processed (hence be completed) during the period $[S_{rk}, T_{rk}]$ reserved to an unavailability period h_{rk} on machine $r = mr_{ij}$. Then,

$$w_{tj} = +\infty, \quad \forall t = S_{rk}, .., T_{rk}$$

2.b. Since operation O_{ij} is non-preemptive and/or unavailability period h_{rk} is non-crossable,

$$w_{tj} = +\infty, \quad \forall t = T_{rk} + 1, .., T_{rk} + p_{ij} - 1$$

6.2.2.4 Adding new columns: Dynamic Programming

For every job J_i, we are searching for a schedule s verifying [6.11]. We would like to find a sequence for which the sum of the contributions of all the job operations $\sum_{j=1}^{n_i} w_{t_j j}$ is minimal, where t_j is the completion period of O_{ij} and $t_{j-1} < t_j - p_{ij} + 1$. And we will add this schedule to the formulation if this sum is strictly smaller than $\gamma^*_{J_i(s)}$. indeed, if there is no such schedule, then no new column needs to be added.

For this, we denote by W_{tj} the contribution of the first j operations of job J_i:
$W_{tj} = \sum_{j'=1}^{j} w_{t_{j'} j'}$
This leads to use the following recurrence formula for Dynamic Programming:

$$W_{tj} = w_{tj} + min_{b_l \leq l < t - p_{ij} + 1} W_{l(j-1)} \tag{6.12}$$

where $b_l = \sum_{j'=1}^{j-1} p_{ij'}$

Note that:

- $W_{t1} = w_{t1}$
- if $w_{tj} = +\infty$ then $W_{tj} = +\infty$

The contributions are stored in a table W of T rows and n_i columns. Hence to calculate W_{tj}, all elements of the table between rows 1 and $t-1$ and columns 1 and $(j-1)$ must be considered.

It can also be represented by a graph $G_{DP} = (N_{DP}, A_{DP})$ where:

- N_{DP} is the set of nodes. A node t_j models operation j and its completion period t. The node weight is w_{tj}.
- A_{DP} is the set of arcs. There exists an arc only between two successive levels (satisfaction of the job routing). There is an arc between periods t_1 and t_2 $(t_2 > t_1)$ if the processing of operation $(j-1)$ ends at period t_1 whereas the processing of operation j ends at period t_2. The arc weight is $W_{t_1(j-1)}$.

The problem consists in searching the shortest path from level 1 to level n_j.

The dynamic programming algorithm 11 allows the improving sequences (columns) through the schedule length to be added. Note that, for each period t, only one column is defined and

Algorithm 11 Dynamic programming algorithm

Begin
  for each job J_i
    for each operation j of the job
      for each period t in the schedule length
        calculate w_{tj}
        calculate W_{tj}
      end for
    end for
    (*) **for** each period t in the schedule length
      if $W_{tn_i} < \gamma_i$
        define the path from the last to the first operations of the job
        Add the column corresponding to that path to RMP
      end if
    end for
  end for
end

added although there might be other optimal paths from the last to the first operations of the job.

When a fixed number of columns are to be added, at each iteration of the column generation process, instruction (*) in the algorithm can be transformed from a *for loop* to a *while loop*. Obviously this takes more computational time; however, when the added columns are sufficiently relevant, RMP will not be overloaded of non necessary columns; which is useful when a branch-and-bound procedure is performed after column generation.

6.2.3 Column generation algorithm

Column generation always works in the feasible domain. Indeed, all the added columns should correspond to feasible schedules for the jobs; and are based on the initial solution that is feasible for the problem. The pricing problem generates a column with a positive reduced cost corresponding to a variable in the primal problem that enters the basis. It is then not necessary to solve the pricing problem to optimality, any column with positive reduced cost can be used. During the column generation process, (RMP) keeps growing. Consequently, if the objective function value of the pricing problem is less than or equal to zero, then the current optimal solution for (RMP) is also optimal for the Master Problem.

The main steps of the column generation algorithm are described below.

From one of the heuristics presented in Chapter 5, an initial feasible solution to the problem is determined. It consists of one schedule or more. From the job sequences associated to this solution, the restricted master problem (RMP) is constructed.

Then, iteratively, the linear relaxation of the (RMP) is solved to find the dual values that

are necessary to define the sequences (columns) that improve the objective value of the primal problem. These sequences are determined by the dynamic programming algorithm and added to (RMP). The process stops when no new column is added.

If the corresponding solution is integral, an optimal solution to (RMP) is obtained, otherwise a branch-and-bound procedure is performed.

The important elements to study in this approach are:

- The quality of the added columns and the number of columns to add at each iteration,
- The difference between the linear relaxation and the solution of the branch-and-bound. This linear relaxation must be the same whatever the initial solution (s). The larger the schedule length, the lower is the linear relaxation,
- The computational time of the column generation and the computational limit for the branch-and-bound,
- The sensitivity to the initial solution,
- The sensitivity to the schedule length.

The general description of the column generation algorithm is provided by Algorithm 12.

Algorithm 12 Algorithm of the column generation approach

Begin
  Compute an initial feasible solution s to the problem.
  Construct the Restricted Master Program (RMP) with the sequences of s
  repeat
    Solve the linear relaxation of (RMP) to obtain the vector of dual values
    Use a heuristic to determine if there is a sequence (or sequences) which satisfies [6.11]
    if the heuristic does not find any sequence
      Use the dynamic programming Algorithm 1 to determine if there is a sequence (or sequences) which satisfies [6.11]
    if at least such a sequence exists
      add it to (RMP)
  until no sequence is found
  if the solution of (RMP) is integral
    Stop
    else Use the branch-and-bound procedure to solve (RMP)
end

In this algorithm, a heuristic is used prior to the dynamic programming algorithm. This algorithm is only used if the heuristic does not find columns to add to (RMP). The goal is to avoid the computational effort of the dynamic programming algorithm and to avoid adding

too many columns to (RMP). This heuristic can be an adaptation of the dynamic programming algorithm. However, it can be more relevant to directly use the dynamic programming algorithm.

6.2.4 Implementation and numerical results

In the following, we will refer to the dynamic programming procedure that finds multiple improving columns per job, at each iteration as $typecol_1$; and to the dynamic programming procedure that finds one improving column per job at each iteration as $typecol_2$.

To evaluate the performance of the column generation model, we compare its computational results to those of the disjunctive model proposed in Chapter 4.

Table 6.1: Test results for disjunctive model for initial benchmarks.

Problem	Disjunctive model				
	Linear relaxation	Integer resolution			
	Lower bound	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	2190	3652	3652	0	0.04
5m5n2	2650	4669	4669	0	0.01
5m5n3	2520	4008	4008	0	0.04
5m5n4	2600	4584	4584	0	0.06
5m5n5	2530	4663	4663	0	0.07
5m10n1	4760	9850	9850	0	672.04
5m10n2	4990	10637	10637	0	167.89
5m10n3	5210	10809.40	11072	2.37	3600
5m10n4	5270	10630	10630	0	565
5m10n5	4960	9706	9706	0	1566.81
10m10n1	10510	15629.28	16878	7.40	3600
10m10n2	10370	15777	15777	0	547.82
10m10n3	10050	15352	15352	0	646.60
10m10n4	9540	14715	15374	4.29	3600
10m10n5	10010	15625	15625	0	1237.39
10m15n1	14880	20750.43	27199	23.71	3600
10m15n2	15440	22652.38	29186	22.39	3600
10m15n3	14740	u			
10m15n4	14990	20860.50	27423	23.93	3600
10m15n5	15530	21465.87	28609	24.97	3600

u: unknown

Table 6.2: Test results for Column Generation model when adding multiple improving columns per job at each iteration in case of initial benchmarks, one initial solution and different schedule lengths.

Problem	Initial solution	Schedule length	Column Generation Model (multiple improving columns per job at each iteration)							
			Linear Relaxation				Branch-and-bound			
			Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	4327	934	3664.5	30	11890	121.01	3856	3856	0	2930.63
	4327	1500	3590.75	12	8371	11.70	**3672**	**3672**	0	704.34
5m5n2	5347	1202	4561	8	5021	4.34	5084	5084	0	197.57
	5347	1500	4561	8	6115	5.84	**4934**	**4934**	0	809.60
5m5n3	6053	1313	3953.33	12	8089	9.75	4082.58	5081	21.54	3600
	6053	1500	3915.79	15	9945	13.85	**4008***	**4008***	0	32.32
5m5n4	4842	1254	4576	21	11825	46.23	**4591**	**4591**	0	25.95
	4842	1500	4576	17	8243	17.64	4618	4618	0	26.23
5m5n5	5536	1391	4493.2	10	8910	9.46	**4868**	**4868**	0	1738.38
	5536	1500	4493.2	12	8433	9.46	4869	4869	0	3600
5m10n1	13375	1686	OFM							
	13375	2000	OFM							
5m10n2	13812	1634	OFM							
	13812	2000	OFM							
5m10n3	15146	1961	OFM							
	15146	2000	OFM							
5m10n4	14700	1888	OFM							
	14700	2000	OFM							
5m10n5	12870	1845	OFM							
	12870	2000	OFM							
10m10n1	19991	2459	OFM							
	19991	2500	OFM							
10m10n2	20881	2315	OFM							
	20881	2500	OFM							
10m10n3	19262	2061	OFM							
	19262	2500	OFM							
10m10n4	19315	2463	OFM							
	19315	2500	OFM							
10m10n5	19322	2404	OFM							
	19322	2500	OFM							
10m15n1	35917	2878	OFM							
	35917	3000	OFM							
10m15n2	36986	2869	OFM							
	36986	3000	OFM							
10m15n3	33121	2774	OFM							
	33121	3000	OFM							
10m15n4	34244	2661	OFM							
	34244	3000	OFM							
10m15n5	36390	2841	OFM							
	36390	3000	OFM							

*: optimal solution

bold: best value

OFM: Out of memory

Table 6.3: Test results for Column Generation model when adding one improving column per job at each iteration in case of initial benchmarks, one initial solution and different schedule lengths.

Problem	Initial solution	Schedule length	Column Generation Model (One improving column per job at each iteration)							
			Linear Relaxation				Branch-and-bound			
			Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1	4327	934	3664.5	318	1203	18.60	4327	4327	0	10.01
	4327	1500	3590.75	177	763	37.95	**3653**	**3653**	0	1.76
5m5n2	5347	1202	4561	201	681	16.90	**4706**	**4706**	0	1.34
	5347	1500	4561	179	663	30.65	4766	4766	0	6.01
5m5n3	6953	1313	3953.33	116	479	11.34	**6053**	**6053**	0	6.04
	6953	1500	3915.79	147	556	23.82	**6053**	**6053**	0	28.06
5m5n4	4282	1254	4576	301	1212	35.76	**4584***	**4584***	0	0.56
	4282	1500	4576	310	1283	64.39	4676	4676	0	17.65
5m5n5	5536	1391	4493.2	318	1352	45.42	4915	4915	0	113.07
	5536	1500	4493.2	228	930	40.73	**4663***	**4663***	0	12.37
5m10n1	13375	1686	OFM							
	13375	2000	OFM							
5m10n2	13812	1634	OFM							
	13812	2000	OFM							
5m10n3	15146	1961	OFM							
	15146	2000	OFM							
5m10n4	14700	1888	OFM							
	14700	2000	OFM							
5m10n5	12870	1845	OFM							
	12870	2000	OFM							
10m10n1	19991	2459	OFM							
	19991	2500	OFM							
10m10n2	20881	2315	OFM							
	20881	2500	OFM							
10m10n3	19262	2061	OFM							
	19262	2500	OFM							
10m10n4	19315	2463	OFM							
	19315	2500	OFM							
10m10n5	19322	2404	OFM							
	19322	2500	OFM							
10m15n1	35917	2878	OFM							
	35917	3000	OFM							
10m15n2	36986	2869	OFM							
	36986	3000	OFM							
10m15n3	33121	2774	OFM							
	33121	3000	OFM							
10m15n4	34244	2661	OFM							
	34244	3000	OFM							
10m15n5	36390	2841	OFM							
	36390	3000	OFM							

*: optimal solution

italic: near optimal solution

bold: best value

OFM: Out of memory

Table 6.1 summarizes the test results for the disjunctive formulation. The first column is the name of the benchmark which is of type $XmYnZ$, where X, Y and Z are respectively the number of machines, number of jobs and number of the benchmarks in the class. Column 2 gives the lower bound of the linear relaxation of the model; whereas Columns 3 to 6 give the results for the mixed integer linear program. Column 3 corresponds to the best lower bound, Column 4 to the objective function of the best solution, Column 5 to the gap (expressed in %), and Column 6 to the CPU time (expressed in seconds).

Table 6.2 (resp. 6.3) shows the test results for the Column Generation model when adding multiple improving columns per job at each iteration in case of initial benchmarks, one initial solution and different schedule lengths. Column 1 is the name of the instance; Column 2 shows the initial solution; Column 3 is the schedule length. Columns 4 to 7 correspond to the test results of the linear relaxation of the restricted master problem. Columns 8 to 11 correspond to the test results of the branch-and-bound procedure. Column 4 shows the lower bound, Column 5 the number of iterations, Column 6 the number of columns, and Column 7 the CPU time (in seconds). Column 8 shows the lower bound, Column 9 the objective function of the best solution, Column 10 the gap (in %), and Column 11 the CPU time (seconds). For each instance, two different schedule lengths are tested for the same initial solution.

From Tables 6.1, 6.2 and 6.3, we deduce that the disjunctive model is more efficient than the Column Generation one for the integer resolution. Indeed, when the disjunctive model is able to almost solve to optimality the instances up to class (10,10) and to provide feasible solution to bigger instances, the column generation model provides a solution to class (5,5) only. However, for the class (5,5) some of the solutions found by the column generation model are close or equal to those found by the disjunctive model. In addition, the column generation model is more efficient than the disjunctive one when providing a lower bound to the linear relaxation. Indeed, the lower bound of the column generation model is better than the disjunctive model one and is close to the best integer solution found.

Tables 6.2 and 6.3 show that the results of the column generation model depend on the value of the schedule length. The larger the schedule length, the lower the lower bound until reaching a stability threshold. Indeed, there is a stability of the value of the lower bound for levels of the schedule length. This can be explained by the fact that, when increasing the value of the schedule length, the set of feasible solutions becomes larger. When the schedule is smaller than the makespan value of the best solutions found for the disjunctive model, the lower bound of the column generation model is larger than these best solutions (see instance 5m5n1).

In general, the column generation considerably improves the values of the objective functions. Starting from the initial solutions, the columns that are added are relevant. However, for the solutions that are improved when performing the column generation process, and whose status at the end of the branch-and-bound procedure is optimal (resp. feasible), we deduce that the added columns are not sufficiently good (resp. sufficiently relevant) to obtain the optimal solution for the global problem.

Also, the computational times are small. This can be also useful when performing the column generation procedure at each node of a branch-and-price method. We could expect that, when increasing the schedule length as the problem induces more schedules so more variables (columns) to consider, the computational time will systematically increase. This is not always true and it can be the opposite. This can be explained by the fact that, when performing the column generation process, more relevant columns are added to the model and make the search easier. However, when the search is longer, it means that the process deals with too many columns.

From these test results, we cannot really deduce which of cases $typecol_1$ and $typecol_2$ is the best. For the linear relaxation part, $typecol_1$ is performed in less iterations but more columns are added. Also, for $typecol_1$, when the search time is longer, too many columns are added and the search is complicated. When the search time is small, there are too many columns but there exist relevant ones and the search is easy. For $typecol_2$, when the search time is longer, the few added columns are not so relevant, the search is then complicated. When the search time is small, the few added columns are relevant and the search is easy. When we compare the best solution over the schedule length provided by cases $typecol_1$ and $typecol_2$ for four instances of five; however for given schedules $typecol_1$ provide better solutions than the ones obtained by $typecol_2$.

To investigate the performance of the column generation approach, the benchmarks are modified by dividing by 10 (scale /10) the schedule length and the durations and the dates of operations and unavailability periods. We ensure to get integer values by rounding non-integer values obtained after dividing by 10 to the immediately upper integer value. This implies rounded values of the objective function of some solutions. The names are then of type $XmYnZscale$, where X, Y, Z and $scale$ are respectively the number of machines, the number of jobs, the number of the benchmarks in the class and the scale. These modifications lead to problems that induces less columns to consider in the column generation approach and less time periods.

Table 6.4 is organized as Table 6.1. It gathers the test results of the disjunctive model. Tables 6.5 and 6.6 (resp. 6.7 and 6.8) show the test results for the column generation model when applying $typecol_1$ (resp. $typecol_2$) adding multiple improving columns (resp. one improving column) per job at each iteration in case of initial benchmarks, one schedule length and different initial solutions. The first table summarizes the results for classes (5,5) and (5,10); whereas the second one presents the results for classes (10,10) and (10,15). Column 1 is the name of the instance; Column 2 is the schedule length; Columns 3 and 4 show the initial solution number and the initial solution. Columns 5 to 8 correspond to the test results of the linear relaxation of the restricted master problem. Columns 9 to 12 correspond to the test results of the branch-and-bound procedure. Column 5 shows the lower bound, Column 6 the number of iterations, Column 7 the number of columns, and Column 8 the CPU time (in seconds). Column 9 shows the lower bound, Column 10 the objective function of the best solution, Column 11 the gap (in %), Column 12 the CPU time (in seconds). For $typecol_1$ (resp. $typecol_2$) and for each instance,

four (resp. three) different solutions are tested for the same schedule length. Note that, for $typecol_1$, we add a fourth solution that is a combination of the three others as it appears that $typecol_1$ is more efficient than $typecol_2$.

Although the modification of the benchmarks has no influence on the disjunctive model, it highly influences the column generation model. Indeed, the column generation model provides solutions to instances up to class (10,15) for $typecol_1$ case and up to class (10,10) for $typecol_2$. The solutions obtained by $typecol_1$ are better than those of $typecol_2$ except for instances 3 and 4 of Class (5,5) for which they are equal and instances of Class (10,10) that are not improved for the two types by the branch-and-bound.

Here also it appears that the quality of the lower bounds of the linear relaxation of the column generation model is excellent. Indeed, their values are close to the optimal solutions. Moreover, these linear relaxations are obtained in short time. The values of the initial solutions are improved through the branch-and-bound procedure for almost all the instances up to (5,10); but for benchmarks of class (10,10) or larger the branch-and-bound is not efficient. However less solutions are improved for $typecol_1$ than $typecol_2$.

For the column generation model and for the same schedule length several initial solutions were tested. We deduce that a lower value of the initial solution does not systematically imply a better solution to the integer resolution. Hence a "good" initial solution does not mean a good solution in terms of the value but in terms of capability to generate relevant columns. Also, we could expect that introducing many feasible schedules generated by heuristics to the initial solution gives a better solution than considering each schedule separately. Unfortunately, it is not always the case. Sometimes it degrades the value of the solution of the branch-and-bound or makes the problem more complicated to solve (even a feasible solution is not provided in one hour) for two instances 3 and 4 of Class (5,10).

Table 6.4: Test results for disjunctive model for modified benchmarks (scale /10).

Problem	Disjunctive model				
	Linear relaxation	Integer resolution			
	Lower bound	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div10	219	367	367	0	0.93
5m5n2div10	265	470	470	0	0.03
5m5n3div10	252	401	401	0	0.07
5m5n4div10	260	460	460	0	0.04
5m5n5div10	253	470	470	0	0.10
5m10n1div10	476	922.09	974	5.33	3600
5m10n2div10	499	1055	1055	0	890.64
5m10n3div10	521	1098	1112	1.26	3600
5m10n4div10	527	1059	1059	0	1133.06
5m10n5div10	496	960	960	0	753.75
10m10n1div10	1051	1545.24	1686	8.35	3600
10m10n2div10	1037	1558	1558	0	2455.08
10m10n3div10	1005	1540	1540	0	1352.06
10m10n4div10	954	1453.36	1519	4.32	3600
10m10n5div10	1001	1551	1551	0	897.75
10m15n1div10	1488	2074.22	2768	25.06	3600
10m15n2div10	1544	2189.98	2969	26.24	3600
10m15n3div10	1474	2028.45	2716	25.31	3600
10m15n4div10	1499	2003.79	2681	25.26	3600
10m15n5div10	1553	2125.50	2771	23.29	3600

Table 6.5: Test results for Column Generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /10), different initial solutions and one schedule length - Part 1.

Problem	Schedule length	Initial Solution number	Initial solution	Column Generation Model (multiple improving columns per job at each iteration)							
				Linear Relaxation				Branch-and-bound			
				Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div10	150	1	**407**	359.53	11	612	0.09	374	374	0	0.78
	150	2	417	359.53	11	818	0.10	**368**	**368**	0	0.51
	150	3	409	359.53	14	677	0.12	376	376	0	2.84
	150	4	1,2,3	359.53	14	677	0.12	376	376	0	2.84
5m5n2div10	150	1	510	459.5	9	726	0.07	**473**	**473**	0	0.34
	150	2	571	459.5	10	767	0.09	480	480	0	0.71
	150	3	**474**	459.5	11	632	0.07	474	474	0	0.09
	150	4	1,2,3	459.5	10	558	0.07	474	474	0	0.09
5m5n3div10	150	1	560	387.33	10	876	0.09	**401***	**401***	0	0.15
	150	2	522	387.33	10	787	0.09	**401***	**401***	0	0.07
	150	3	**488**	387.33	12	842	0.10	**401***	**401***	0	0.15
	150	4	1,2,3	387.33	11	876	0.15	**401***	**401***	0	0.15
5m5n4div10	150	1	585	459.4	13	762	0.12	**463**	**463**	0	0.09
	150	2	**546**	459.4	11	813	0.12	**463**	**463**	0	0.07
	150	3	572	459.4	12	988	0.14	**463**	**463**	0	0.40
	150	4	1,2,3	459.4	13	706	0.10	**463**	**463**	0	0.23
5m5n5div10	150	1	569	446.31	10	720	0.09	**478**	**478**	0	2.60
	150	2	575	446.31	10	777	0.09	**478**	**478**	0	1.56
	150	3	**551**	446.31	9	655	0.07	494	494	0	2.51
	150	4	1,2,3	446.31	9	707	0.07	**478**	**478**	0	3.62
5m10n1div10	200	1	1439	937	25	5452	3.21	937.78	1073	12.72	3600
	200	2	1354	937	21	4724	2.81	937.83	**1043**	10.18	3600
	200	3	**1214**	937	26	5001	2.96	937.82	1122	16.56	3600
	200	4	1,2,3	937	27	4402	2.62	937.71	1075	12.89	3600
5m10n2div10	200	1	1380	980.77	34	6191	5.73	981.56	**1138**	13.87	3600
	200	2	1365	980.77	42	6615	6.17	981.48	1167	16.03	3600
	200	3	**1322**	980.77	34	6226	5.07	981.55	1145	14.40	3600
	200	4	1,2,3	980.77	30	6307	4.54	981.53	1170	16.25	3600
5m10n3div10	200	1	**1420**	1067.55	20	4287	1.84	1070.47	1271	15.90	3600
	200	2	1539	1067.55	25	4584	2.04	1069.63	**1159**	7.78	3600
	200	3	1430	1067.55	21	4251	2.14	1070.03	1269	15.80	3600
	200	4	1,2,3	1067.55	21	4282	1.64	u			3600
5m10n4div10	200	1	1352	1031.62	25	4399	2.25	1033.35	1352	23.74	3600
	200	2	**1309**	1031.62	34	4574	2.78	1032.88	**1191**	13.39	3600
	200	3	1476	1031.62	24	4294	2.09	1034.16	1367	24.53	3600
	200	4	1,2,3	1031.63	20	4469	1.82	u			3600
5m10n5div10	200	1	1375	887.35	31	5739	5.78	887.79	1375	35.69	3600
	200	2	**1207**	887.35	31	5975	6.06	887.91	1117	20.69	3600
	200	3	1270	887.35	29	5617	5.01	887.87	1270	30.33	3600
	200	4	1,2,3	887.35	29	5996	6.10	887.78	**1115**	20.56	3600

bold: best value

*: optimal solution

italic: near optimal solution

u: unknown

Table 6.6: Test results for Column Generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /10), different initial solutions and one schedule length - Part 2.

Problem	Schedule length	Initial Solution number	Initial solution	Column Generation Model (multiple improving columns per job at each iteration)							
				Linear Relaxation				Branch-and-bound			
				Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
10m10n1div10	250	1	2144	1523.3	72	10275	38.26	1523.63	2144	29.07	3600
	250	2	**2040**	1523.3	53	11145	37.23	1523.63	**2040**	25.44	3600
	250	3	2071	1523.3	57	10459	35.68	1523.62	2071	26.56	3600
	250	4	1,2,3	1523.3	58	10086	34.82	1523.63	2144	29.07	3600
10m10n2div10	250	1	2152	1475.36	34	8955	32.25	1476.85	2152	31.52	3600
	250	2	1988	1475.36	49	10338	40.59	1476.37	1988	26.24	3600
	250	3	**1907**	1475.36	44	9388	33.39	1476.25	**1907**	22.71	3600
	250	4	1,2,3	1475.36	52	9765	35.32	1477.56	2142	31.49	3600
10m10n3div10	250	1	1944	1468.28	33	8713	24.70	1469	1944	24.56	3600
	250	2	**1822**	1468.28	32	9296	24.85	1469.16	**1822**	19.47	3600
	250	3	2053	1468.28	31	9769	25.12	1468.84	2053	28.59	3600
	250	4	1,2,3	1468.28	31	9217	24.17	1469.09	1944	24.56	3600
10m10n4div10	250	1	1939	1384.6	31	9815	28.48	1385.29	1939	28.70	3600
	250	2	1914	1384.6	31	9971	29.03	1385.04	1914	27.77	3600
	250	3	**1883**	1384.6	33	10176	29.23	1385.12	**1883**	26.58	3600
	250	4	1,2,3	1384.6	34	10090	27.26	1385.16	1939	28.71	3600
10m10n5div10	250	1	1958	1443.48	36	9432	35.20	1443.80	1958	26.40	3600
	250	2	**1801**	1443.48	38	9632	33.85	1443.04	**1801**	19.93	3600
	250	3	1877	1433.48	35	9086	29.29	1443.93	1877	23.20	3600
	250	4	1,2,3	1443.48	41	9245	34.01	1444.01	1958	26.39	3600
10m15n1div10	300	1	**3429**	2321.09	77	34833	1082.66	2321.09	**3429**	32.45	3600
	300	2	3507	2321.09	85	34534	1029.75	2321.09	3507	33.96	3600
	300	3	3494	2321.09	77	34766	1042.19	2321.12	3494	33.71	3600
	300	4	1,2,3	2321.09	80	36269	1139.97	2321.10	**3429**	32.45	3600
10m15n2div10	300	1	3880	2630.76	38	21166	263.89	2630.76	3880	32.32	3600
	300	2	3848	2630.76	43	21061	2615.76	2630.76	3848	31.76	3600
	300	3	**3698**	2630.76	40	21664	254.71	2630.76	**3698**	28.98	3600
	300	4	1,2,3	2630.76	39	22386	256.34	2630.76	3880	32.32	3600
10m15n3div10	300	1	3524	2403.45	53	25862	387.12	2403.45	3524	31.93	3600
	300	2	3693	2403.45	55	26278	355.25	2403.50	3693	35.06	3600
	300	3	**3474**	2403.45	63	27179	387.17	2403.50	**3474**	30.95	3600
	300	4	1,2,3	2403.45	64	28235	389.65	2403.60	3693	34.78	3600
10m15n4div10	300	1	3583	2341.9	43	27458	581.81	2341.90	3583	34.78	3600
	300	2	3437	2341.9	43	28186	584.09	2341.90	3437	32.00	3600
	300	3	**3348**	2341.9	44	27433	582.12	2341.90	**3348**	30.19	3600
	300	4	1,2,3	2341.9	47	28199	608.70	2341.91	3583	34.78	3600
10m15n5div10	300	1	3520	2436.32	74	34924	1055.28	2436.32	3520	30.92	3600
	300	2	3590	2436.32	74	37271	1118.23	2436.33	3590	32.27	3600
	300	3	**3439**	2436.32	74	35781	1004.11	2436.32	**3439**	29.28	3600
	300	4	1,2,3	2436.32	81	36931	1065.73	2436.33	3520	30.92	3600

bold: best value

Table 6.7: Test results for Column Generation model when adding one improving column per job at each iteration in case of modified benchmarks (scale /10), different initial solutions and one schedule length - Part 1.

Problem	Schedule length	Initial Solution number	Initial solution	Column Generation Model (One improving column per job at each iteration)							
				Linear Relaxation				Branch-and-bound			
				Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div10	150	1	**407**	359.53	50	232	0.23	**369**	**369**	0	0.53
	150	2	417	359.53	61	247	0.28	371	371	0	0.54
	150	3	409	359.53	53	208	0.23	**369**	**369**	0	0.26
5m5n2div10	150	1	510	459.5	47	199	0.18	**474**	**474**	0	0.14
	150	2	571	459.5	41	171	0.17	480	480	0	0.12
	150	3	**474**	459.5	29	105	0.10	**474**	**474**	0	0.03
5m5n3div10	150	1	560	387.33	40	181	0.17	475	475	0	0.62
	150	2	522	387.33	65	307	0.28	472	472	0	1.18
	150	3	**488**	387.33	35	150	0.14	**401***	**401***	0	0.04
5m5n4div10	150	1	585	459.4	98	407	0.51	469	469	0	0.39
	150	2	**546**	459.4	82	348	0.35	**463**	**463**	0	0.04
	150	3	572	459.4	90	366	0.42	465	465	0	0.18
5m5n5div10	150	1	569	446.31	53	250	0.23	**498**	**498**	0	0.53
	150	2	575	446.31	48	208	0.20	508	508	0	0.51
	150	3	**551**	446.31	56	242	0.23	508	508	0	0.67
5m10n1div10	200	1	1439	937	441	3253	20.23	997	**1037**	2.90	3600
	200	2	1354	937	400	2873	16.58	1354	1354	0	3480.05
	200	3	**1214**	937	466	3616	23.39	937.63	1058	11.48	3600
5m10n2div10	200	1	1380	980.77	613	5075	46.20	1370	1370	0	2283.83
	200	2	1365	980.77	499	4145	30.14	1167	**1167**	0	1336.03
	200	3	**1322**	980.77	477	4179	30.14	981.62	1310	25.26	3600
5m10n3div10	200	1	**1420**	1067.55	261	2245	9.96	1187	**1187**	0	267.39
	200	2	1539	1067.55	272	2429	9.96	1529	1529	0	3348.97
	200	3	1430	1067.55	347	3149	19.39	1069.36	1430	25.40	3600
5m10n4div10	200	1	1352	1031.62	404	2973	16.64	1352	1352	0	1986.88
	200	2	**1309**	1031.62	681	4595	1021.62	1034.27	**1279**	19.09	3600
	200	3	1476	1031.62	408	2892	13.85	1034.35	1476	30.13	3600
5m10n5div10	200	1	1375	887.35	592	4958	46	918.87	1375	33.45	3600
	200	2	**1207**	887.35	650	5453	57.35	887.81	**1206**	26.60	3600
	200	3	1270	887.35	600	4756	43.14	887.64	1270	30.35	3600

bold: best value

*: optimal solution

Table 6.8: Test results for Column Generation model when adding one improving column per job at each iteration in case of modified benchmarks (scale /10), different initial solutions and one schedule length - Part 2.

Problem	Schedule length	Initial Solution number	Initial solution	Column Generation Model (One improving column per job at each iteration)							
				Linear Relaxation				Branch-and-bound			
				Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
10m10n1div10	250	1	2144	1523.3	885	6931	224.89	1524.54	2144	29.03	3600
	250	2	**2040**	1523.3	1176	9282	446.73	1523.70	**2040**	25.43	3600
	250	3	2071	1523.3	1060	7656	259.14	1523.72	2071	26.55	3600
10m10n2div10	250	1	2152	1475.36	777	6714	245.14	1478.29	2152	31.45	3600
	250	2	1988	1475.36	1141	9866	600.23	1477.38	1988	26.19	3600
	250	3	**1907**	1475.36	713	6487	230.06	1477.71	**1907**	22.63	3600
10m10n3div10	250	1	1944	1468.28	957	7780	385.62	1473.23	1944	24.34	3600
	250	2	**1822**	1468.28	828	6799	264.43	1468.80	**1822**	19.49	3600
	250	3	2053	1468.28	950	7376	305.92	1468.93	2053	28.59	3600
10m10n4div10	250	1	1939	1384.6	908	8149	346.20	1386	1939	28.67	3600
	250	2	1914	1384.6	948	8515	341.18	1385.03	1914	27.78	3600
	250	3	**1883**	1384.6	906	8418	345.26	1385.11	**1883**	26.58	3600
10m10n5div10	250	1	1958	1443.48	1147	9421	414.28	1444.11	1958	26.38	3600
	250	2	1801	OFM							
	250	3	**1877**	1443.48	905	7855	348.45	1443.95	**1877**	23.19	3600
10m15n1div10	300	1	3429	OFM							
	300	2	3507	OFM							
	300	3	3494	OFM							
10m15n2div10	300	1	3880	OFM							
	300	2	3848	OFM							
	300	3	3698	OFM							
10m15n3div10	300	1	3524	OFM							
	300	2	3693	OFM							
	300	3	3474	OFM							
10m15n4div10	300	1	3583	OFM							
	300	2	3437	OFM							
	300	3	3348	OFM							
10m15n5div10	300	1	3520	OFM							
	300	2	3590	OFM							
	300	3	3439	OFM							

bold: best value

OFM: Out of memory

Tables 6.9 and 6.10 show the test results for the column generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /10), one initial solution and different schedule lengths as $typecol_1$ gives better results than $typecol_2$. They are organized as Table 6.2 except that positions of the initial solution and the schedule length are permuted. Hence, Columns 2 shows the initial solution and Column 3 the schedule length.

For the previous test results, these results show that modifying the schedule length has an impact on the lower bound of the linear relaxation. Indeed, when the lower bound is not the same, it decreases due to the increase of the schedule length until reaching a stability value. The number of columns increases with the increase of the schedule length.

We could expect that increasing the schedule length will complicate the problem due to the huge number of variables (columns) in the master problem. But, for all the instances, the column generation model provides a solution and sometimes this solution is better than others given by a lower value of the schedule length.

Table 6.9: Test results for Column Generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /10), one initial solution and different schedule lengths - Part 1.

Problem	Initial solution	Schedule length	Column Generation Model (multiple improving columns per job at each iteration)							
			Linear Relaxation				Branch-and-bound			
			Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div10	407	150	359.53	11	612	0.09	374	374	0	0.78
	407	**105**	363.5	10	526	0.06	374	374	0	0.26
	407	200	359.53	10	624	0.12	**368**	**368**	0	0.5
	407	400	359.53	10	759	0.34	369	369	0	0.87
5m5n2div10	510	150	459.5	9	726	0.07	473	473	0	0.34
	510	**110**	467	11	459	0.06	510	510	0	0.14
	510	200	459.5	12	912	0.15	**471**	**471**	0	0.42
	510	400	459.5	9	918	0.31	472	472	0	0.39
5m5n3div10	560	150	387.33	10	876	0.09	**401***	**401***	0	0.15
	560	**137**	387.33	9	838	0.07	**401***	**401***	0	0.20
	560	200	387.33	13	1176	0.17	**401***	**401***	0	0.26
	560	400	387.33	11	1735	0.40	427	427	0	1.90
5m5n4div10	585	150	459.4	13	762	0.12	463	463	0	0.09
	585	**139**	459.4	11	699	0.10	463	463	0	0.32
	585	200	459.4	11	920	0.15	463	463	0	0.25
	585	400	459.4	11	1396	0.48	**460***	**460***	0	0.07
5m5n5div10	569	150	446.31	10	720	0.09	**478**	**478**	0	2.60
	569	**134**	446.31	9	639	0.06	**478**	**478**	0	0.93
	569	200	446.31	9	899	0.12	**478**	**478**	0	2.07
	569	400	446.31	11	1431	0.42	**478**	**478**	0	1.85
5m10n1div10	1439	200	937	25	5452	3.21	937.78	**1073**	12.72	3600
	1439	**180**	937	26	5454	3.48	937.58	1083	13.55	3600
	1439	250	937	28	5928	3.95	937.85	1089	14.01	3600
	1439	450	937	24	6835	5.54	937.75	1147	18.40	3600
5m10n2div10	1380	200	980.77	34	6191	5.73	981.56	**1138**	13.87	3600
	1380	**164**	981.10	36	6134	5	981.87	1380	29.06	3600
	1380	250	980.77	39	7164	6.87	981.54	1141	14.10	3600
	1380	450	980.77	37	8499	8.78	981.66	1194	17.93	3600
5m10n3div10	1420	200	1067.55	20	4287	1.84	1070.47	1271	15.90	3600
	1420	**189**	1067.55	20	4325	1.82	1070.90	1420	24.76	3600
	1420	250	1067.55	19	4290	1.84	1068.54	**1194**	10.60	3600
	1420	450	1067.55	24	6348	4.10	1068.72	1351	21.05	3600
5m10n4div10	1352	200	1031.62	25	4399	2.25	1033.35	1352	23.74	3600
	1352	**186**	1031.74	26	4414	2.10	1032.59	1352	23.80	3600
	1352	250	1031.62	27	4969	2.59	1034.63	**1209**	14.54	3600
	1352	450	1031.62	25	6015	4.09	1034.70	1277	19.12	3600
5m10n5div10	1375	200	887.35	31	5739	5.78	887.79	1375	35.69	3600
	1375	**181**	887.35	32	5679	5.53	887.81	**1074**	17.50	3600
	1375	250	887.35	30	6096	6.46	887.88	1128	21.48	3600
	1375	450	887.35	34	7449	9.10	887.89	1163	23.86	3600

bold: best value

*: optimal solution

italic: near optimal solution

Table 6.10: Test results for Column Generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /10), one initial solution and different schedule lengths - Part 2.

Problem	Initial solution	Schedule length	Column Generation Model (multiple improving columns per job at each iteration)							
			Linear Relaxation				Branch-and-bound			
			Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
10m10n1div10	2144	250	1523.3	72	10275	38.26	1523.63	**2144**	29.07	3600
	2144	**242**	1523.3	47	10151	32.85	1523.59	**2144**	29.07	3600
	2144	300	1523.3	57	10991	38.59	1523.67	**2144**	29.07	3600
	2144	500	1523.3	51	11142	43.89	1523.68	**2144**	29.07	3600
10m10n2div10	2152	250	1475.36	34	8955	32.25	1476.85	**2152**	31.52	3600
	2152	**247**	1475.36	41	9316	34.48	1476.88	**2152**	31.52	3600
	2152	300	1475.36	42	9450	33.95	1477.06	**2152**	31.52	3600
	2152	500	1475.36	42	11571	38.15	1479.03	**2152**	31.51	3600
10m10n3div10	1944	250	1468.28	33	8713	24.70	1469	**1944**	24.56	3600
	1944	**243**	1468.28	32	8853	25.81	1468.86	**1944**	24.57	3600
	1944	300	1468.28	35	9261	26.37	1468.87	**1944**	34.57	3600
	1944	500	1468.28	32	11571	38.15	1459.03	**1944**	20.45	3600
10m10n4div10	1939	250	1384.6	31	9815	28.48	1385.29	**1939**	28.70	3600
	1939	**234**	1384.6	34	10122	30.35	1385.17	**1939**	28.71	3600
	1939	300	1384.6	37	10888	31.87	1385.15	**1939**	28.71	3600
	1939	500	1384.6	37	13017	47.59	1375.29	**1939**	28.70	3600
10m10n5div10	1958	250	1443.48	36	9432	35.20	1443.80	1958	26.40	3600
	1958	**233**	1443.48	34	9092	31.17	1443.90	1958	26.39	3600
	1958	300	1443.48	40	10228	42.43	1444.15	1919	24.87	3600
	1958	500	1443.48	42	12225	54.95	1433.92	**1885**	23.52	3600
10m15n1div10	3429	300	2321.09	77	34833	1082.66	2321.09	**3429**	32.45	3600
	3429	**263**	2321.09	76	34243	1045.09	2321.09	**3429**	32.45	3600
	3429	350	2321.09	79	34651	991.23	2321.09	**3429**	32.45	3600
	3429	550	2321.09	78	38794	1132.23	2321.12	**3429**	32.45	3600
10m15n2div10	3880	300	2630.76	38	21166	263.89	2630.76	**3880**	32.32	3600
	3880	**287**	2630.76	40	21925	272.42	2630.76	**3880**	32.32	3600
	3880	350	2630.76	36	21728	263.68	2630.76	**3880**	32.32	3600
	3880	550	2630.76	40	23483	311.78	2630.76	**3880**	32.32	3600
10m15n3div10	3524	300	2403.45	53	25862	387.12	2403.45	3524	31.93	3604
	3524	**262**	2403.45	64	25343	350.58	2403.45	3524	31.93	3600
	3524	350	2403.45	50	26376	381.95	2403.50	3524	31.93	3600
	3524	550	2403.45	54	29347	400.76	2403.50	**3407**	29.58	3600
10m15n4div10	3583	300	2341.9	43	27458	581.81	2341.90	3583	34.78	3600
	3583	**276**	2341.9	43	28058	615.5	2341.90	3583	34.78	3600
	3583	350	2341.9	48	29317	624.57	2341.90	3583	34.78	3600
	3583	500	2341.9	42	30364	648.45	2341.91	**3494**	33.12	3600
10m15n5div10	3520	300	2436.32	74	34924	1055.28	2436.32	**3520**	30.92	3600
	3520	**286**	2436.32	70	35150	956.15	2436.33	**3520**	30.92	3600
	3520	350	2436.32	68	36094	992.95	2436.32	**3520**	30.92	3600
	3520	500	2436.32	72	38668	1059.94	2436.32	**3520**	30.92	3600

bold: best value

Table 6.11: Test results for disjunctive model for modified benchmarks (scale /20).

Problem	Disjunctive model				
	Linear relaxation	Integer resolution			
	Lower bound	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div20	114	187	187	0	0.07
5m5n2div20	138	236	236	0	0.03
5m5n3div20	135	219	219	0	0.06
5m5n4div20	135	244	244	0	0.06
5m5n5div20	133	248	248	0	0.12
5m10n1div20	248	480	506	5.14	3600
5m10n2div20	260	540	540	0	328.31
5m10n3div20	277	537	582	7.63	3600
5m10n4div20	275	560	560	0	976.04
5m10n5div20	260	506	506	0	636.76
10m10n1div20	556	816	869	6.10	3600
10m10n2div20	548	826	826	0	374.14
10m10n3div20	526	780	780	0	654.03
10m10n4div20	503	762	803	5.11	3600
10m10n5div20	527	822	822	0	3571.27
10m15n1div20	786	1080.82	1442	25.05	3600
10m15n2div20	817	1158.03	1528	24.21	3600
10m15n3div20	778	1063.25	1448	26.57	3600
10m15n4div20	790	1088.29	1378	21.02	3600
10m15n5div20	821	1101.26	1611	31.64	3600

Table 6.12: Test results for Column Generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /20).

Problem	Initial solution	Schedule length	Column Generation Model (multiple improving columns per job at each iteration)							
			Linear Relaxation				Branch-and-bound			
			Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div20	230	75	191.8	7	339	0.01	209	209	0	0.37
5m5n2div20	254	75	232	7	337	0.03	251	251	0	0.18
5m5n3div20	282	75	209.41	14	369	0.03	223	223	0	0.17
5m5n4div20	268	75	241.5	7	241	0.03	254	254	0	0.25
5m5n5div20	289	75	237.64	10	417	0.03	266	266	0	1.59
5m10n1div20	620	100	495.47	16	2178	0.48	551	551	0	3235.90
5m10n2div20	654	100	526.28	22	2586	0.76	579	579	0	2380.61
5m10n3div20	702	100	561.18	15	2125	0.39	683	683	0	1985.86
5m10n4div20	655	100	545.93	16	1979	0.45	625	625	0	1869.28
5m10n5div20	625	100	475.59	19	2329	0.62	477.56	556	14.36	3600
10m10n1div20	1030	150	823.09	30	4641	3.84	823.50	1030	20.24	3600
10m10n2div20	1027	150	797.51	34	3976	3.54	797.83	1027	22.53	3600
10m10n3div20	991	150	767.56	19	4321	2.84	768.09	991	22.72	3600
10m10n4div20	974	150	755.04	25	4257	3.59	755.43	974	22.67	3600
10m10n5div20	978	150	774.59	32	4555	4.26	775.28	978	20.03	3600
10m15n1div20	1684	200	1252.08	56	13957	84.12	1252.21	1684	25.87	3600
10m15n2div20	1827	200	1405.2	20	10401	27.79	1405.48	1827	23.26	3600
10m15n3div20	1757	200	1298.13	32	13155	57.60	1298.28	1757	26.33	3600
10m15n4div20	1684	200	1255.67	35	12012	52.56	1255.83	1684	25.65	3600
10m15n5div20	1767	200	1310.84	40	14404	76.98	1311.07	1767	26.02	3600

Table 6.11 is organized as Table 6.1. It summarizes the test results for the disjunctive model for modified benchmarks (scale /20). Tables 6.12 (6.13) shows the test results for column generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /20), one initial solution and one schedule length. They are organized as Table 6.2 except that positions of the initial solution and the schedule length are permuted. Hence, Column 2 shows the initial solution and Column 3 the schedule length.

The main goal for this benchmark modification is to improve the branch-and-bound results of the benchmarks of classes (10,10) and (10,15). However, the reduction of the scale is not significant. This confirms the results obtained for the time-indexed model of Chapter 4.

The results show that the improved solutions (instances of Classes (5,5) and (5,10)) are very good because the gaps between the disjunctive MIP solutions and and the solutions of the column generation branch-and-bound are small.

Table 6.13: Test results for Column Generation model when adding one improving column per job at each iteration in case of modified benchmarks (scale /20).

Problem	Initial solution	Schedule length	Column Generation Model (One improving column per job at each iteration)							
			Linear Relaxation				Branch-and-bound			
			Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div20	230	75	191.8	23	108	0.04	226	226	0	0.40
5m5n2div20	254	75	232	21	96	00.3	240	240	0	0.01
5m5n3div20	282	75	209.41	28	124	0.06	282	282	0	0.3
5m5n4div20	268	75	241.5	49	221	0.09	268	268	0	0.32
5m5n5div20	289	75	237.64	54	305	0.11	266	266	0	1.75
5m10n1div20	620	100	495.47	125	1015	1.32	620	620	0	69
5m10n2div20	654	100	526.28	204	1829	3.92	654	654	0	117.12
5m10n3div20	702	100	561.18	130	1050	1.40	702	702	0	53.96
5m10n4div20	655	100	545.93	178	1105	1.68	655	655	0	80.10
5m10n5div20	625	100	475.59	164	1289	2	625	625	0	191.60
10m10n1div20	1030	150	823.09	297	2676	16.89	1030	1030	0	1165.61
10m10n2div20	1027	150	797.51	307	2495	16.64	1027	1027	0	852.71
10m10n3div20	991	150	767.56	309	2822	20.82	991	991	0	1018.69
10m10n4div20	974	150	755.04	342	2763	19.35	974	974	0	1002.33
10m10n5div20	978	150	774.59	484	3601	29.15	978	978	0	1542.03
10m15n1div20	1684	200	1252.08	OFM						
10m15n2div20	1827	200	1405.2	OFM						
10m15n3div20	1757	200	1298.13	OFM						
10m15n4div20	1684	200	1255.67	OFM						
10m15n5div20	1767	200	1310.84	OFM						

OFM: Out of memory

6.2.5 Extension: The preemptive case

The approach is similar to the non-preemptive case. The differences are in the expressions of w_{tj}, W_{tj} and in the columns to be added.

Indeed, for a given job J_i, since the contribution of operation O_{ij} in the reduced cost is defined by the values of the dual variables δ_{lr} associated to the processing periods of the operation, $\sum_{l=t-p_{ij}+1}^{t} \delta^*_{lmr_{ij}}$ in expression [6.10], w_{tj} will be replaced by $\sum_{l \in P_{tj}} \delta^*_{lmr_{ij}}$, where:

- $P_{tj} = [t - p_{ij} + 1, t]$ if operation O_{ij} is not interrupted by an unavailability period.

- $P_{tj} = [S_{rk} - p_{ij} - (t - T_{rk}), S_{rk} - 1] \bigcup [T_{rk} + 1, t]$ if O_{ij} is resumable.

The point [2.b.] of Remarks of Section 6.2.2.3 is no longer true, as it is possible to complete an operation just after the end of the interrupting unavailability period.

Numerical results

Table 6.14 summarizes the test results for modified benchmarks (scale /10) with the disjunctive formulation for resumable operations. The first column is the name of the benchmark. Columns 2 to 5 give the results for the mixed integer linear program. Column 2 shows the best lower bound, Column 3 the objective function of the best solution, Column 4 the gap (expressed in %), and Column 5 the CPU time (expressed in seconds).

Table 6.15 shows the test results for the Column Generation model when adding multiple improving columns per job at each iteration, i.e. $typecol_1$, in case of initial benchmarks, one initial solution and one schedule length. Column 1 shows the name of the instance, Column 2 the initial solution and Column 3 the schedule length. Columns 4 to 7 correspond to the test results of the linear relaxation of the restricted master problem. Columns 8 to 11 correspond to the test results of the branch-and-bound procedure. Column 4 shows the lower bound, Column 5 the number of iterations, Column 6 the number of columns, and Column 7 the CPU time (in seconds). Column 8 is the lower bound, Column 9 the objective function of the best solution, Column 10 the gap (in %), Column 11 the CPU time (seconds). For each instance, only one schedule length is tested for one initial solution.

The conclusions of the test results are the same as for the non-preemptive case. These conclusions concern the classes of instances solved, the quality of the solutions and the linear relaxations, the CPU time.

Table 6.14: Test results for resumable disjunctive model for modified benchmarks (scale /10).

Problem	Disjunctive model Integer resolution			
	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div10	359	359	0	0.79
5m5n2div10	434	434	0	0.14
5m5n3div10	391	391	0	1.10
5m5n4div10	413	413	0	0.23
5m5n5div10	446	446	0	1.32
5m10n1div10	771.83	952	18.92	3600
5m10n2div10	842.30	1044	19.32	3600
5m10n3div10	954.33	1073	11.06	3600
5m10n4div10	855.20	1062	19.47	3600
5m10n5div10	821.00	941	12.75	3600
10m10n1div10	1388.00	1651	15.93	3600
10m10n2div10	1358.55	1524	10.86	3600
10m10n3div10	1351.40	1524	11.33	3600
10m10n4div10	1282.32	1481	13.42	3600
10m10n5div10	1362.72	1513	9.93	3600
10m15n1div10	1898.61	2684	29.26	3600
10m15n2div10	1958.83	2932	33.19	3600
10m15n3div10	1851.63	2768	33.11	3600
10m15n4div10	1816.47	2705	32.85	3600
10m15n5div10	1925.00	2825	31.86	3600

Table 6.15: Test results for resumable Column Generation model when adding multiple improving columns per job at each iteration in case of modified benchmarks (scale /10).

Problem	Initial solution	Schedule length	Column Generation Model (multiple improving columns per job at each iteration)							
			Linear Relaxation				Branch-and-bound			
			Lower bound	Number of iterations	Number of columns	CPU (sec)	Best bound	Best solution	Gap (%)	CPU (sec)
5m5n1div10	407	150	345.73	17	1154	0.23	364	364	0	6.62
5m5n2div10	510	150	432.5	15	1063	0.20	434	434	0	0.04
5m5n3div10	560	150	372.66	10	829	0.10	396	396	0	2.28
5m5n4div10	585	150	410.5	11	966	0.17	415	415	0	0.34
5m5n5div10	569	150	421.92	11	798	0.17	459	459	0	23.37
5m10n1div10	1439	200	909.14	29	5394	6.12	909.92	1078	15.74	3600
5m10n2div10	1380	200	964.71	36	6764	10.09	965.00	1226	21.46	3600
5m10n3div10	1420	200	1041.4	18	4070	2.76	1041.88	1300	20.01	3600
5m10n4div10	1352	200	1019.68	20	4696	3.04	1020.50	1352	24.70	3600
5m10n5div10	1375	200	877.01	27	5886	9.28	877.33	1375	36.46	3600
10m10n1div10	2144	250	1501.36	42	11235	71.21	1501.36	2144	30.11	3600
10m10n2div10	2152	250	1437.83	31	9156	50.92	1437.38	2152	33.32	3600
10m10n3div10	1944	250	1422.07	41	10115	47.37	1422.36	1944	26.97	3600
10m10n4div10	1939	250	1357.42	44	10104	63.32	1357.43	1939	30.15	3600
10m10n5div10	1958	250	1387.73	58	10992	72.26	1388.06	1958	29.26	3600
10m15n1div10	3429	300	2276.78	60	31116	1635.38	2276.78	3429	33.75	3600
10m15n2div10	3880	300	2571.12	31	16728	234.31	2556.12	3880	33.86	3600
10m15n3div10	3524	300	2352.56	53	24521	588.04	2352.57	3524	33.38	3600
10m15n4div10	3583	300	2314.48	50	25794	917.95	2314.49	3583	35.55	3600
10m15n5div10	3520	300	2392.68	66	36020	1843.98	2392.68	3520	32.16	3600

6.3 Job shop problem with flexible availability periods on resources

The adapted integer programming formulation, for the job shop problem with flexible resource unavailability periods, is presented in Section 6.3.1. Section 6.3.2 describes the column generation approach; and the main steps are provided in Section 6.3.3.

6.3.1 Adapted integer programming formulation

In addition to the n schedules of the jobs on the machines, the solution to the problem is also represented by the H schedules of the machine unavailability periods. In the following, the unavailability period associated to a given schedule u is denoted by $h(u)$.

Associating schedules to each unavailability period induces less schedules than associating schedules to each machine. Indeed, if we assume that a machine has two unavailability periods and the length of the time window for the first (resp. second) unavailability period is 5 (resp. 3), associating schedules to each unavailability period induces three feasible schedules for the first unavailability period and two for the second one, the total is 5+3; whereas associating schedules to the machine induces 5×3 feasible schedules; this number corresponds to the combinations of positions of the two unavailability periods.

In order to introduce the formulation, we need to add the following parameters:

U: the set of unavailability periods schedules.

We denote by $ES_{h(u)}$ (resp. $LS_{h(u)}$) the earliest (resp. latest) period of unavailability period $h(u)$; and $p'_{h(u)}$ the duration of $h(u)$.

$b_t^u = 1$ if unavailability period $h(u)$ is planned at period t on machine $r(h)$ and 0 otherwise. Note that $b_t^u = 0$ if $t \in [0, ES_{h(u)}[\bigcup]LS_{h(u)} + p'_{h(u)}, T]$.

$U(h)$: set of sequences of unavailability period h. $U(h)_t = 1$ if unavailability period h is processed on period t and 0 otherwise.

We also add the following variable:

$y_u = 1$ if the u^{th} feasible schedule is used for unavailability period $h(u)$ and 0 otherwise.

Hence, the integer program (MP) can be adapted as follows:

$$f^* = \min f = \sum_{s \in S} C_s x_s \tag{6.13}$$

$$\sum_{s \in S(J_i)} x_s = 1 \qquad i = 1, .., n \tag{6.14}$$

$$\sum_{u \in U(h)} y_u = 1 \qquad h = 1, .., H \tag{6.15}$$

$$\sum_{s \in S} \sum_{j=1; m r_{ij} = r}^{n_{J_i(s)}} a_{tj}^s x_s + \sum_{u \in U; m_{h(u)} = r} b_t^u y_u \leq 1 \qquad t = 1, .., T; r = 1, .., m \tag{6.16}$$

$$x_s \in \{0, 1\} \qquad \forall s \in S \tag{6.17}$$

$$y_u \in \{0,1\} \qquad \forall u \in U \tag{6.18}$$

In the sequel, we refer to this model as (aMP) for adapted-(MP). Here also, the objective is the minimization of the sum of the completion dates of the schedules. Constraint [6.14] (resp. [6.15]) ensures that only one schedule is selected for each job (resp. unavailability period). Constraint [6.16] guarantees that, when a machine is available, it is never used by more than one operation at each period.

6.3.2 Adapted column generation approach

6.3.2.1 Master problem

As the objective is to solve the linear relaxation of (aMP), we proceed in the same manner as for the Master Problem of Section 6.2.2, to obtain the *adapted Reduced Master Problem* called (aRMP).

6.3.2.2 Dual problem

Dual variables are obtained for every constraint of the Master Problem, denoted by γ_i for the first n constraints (constraints [6.14]), θ_u for the H constraints (constraints [6.15]) and δ_{tr} for the $T \times m$ last constraints (constraints [6.16]).

Let us consider the following linear program:

$$d^* = \max d = \sum_{i=1}^{n} \gamma_i + \sum_{h=1}^{H} \theta_h - \sum_{t=1}^{T} \sum_{r=1}^{m} \delta_{tr} \tag{6.19}$$

$$\gamma_{J_i(s)} - \sum_{t=1}^{T} \sum_{j=1}^{n_{J_i(s)}} a_{tj}^{s} \delta_{tmr_{ij}} \leq C_s \qquad \forall s \in S \tag{6.20}$$

$$\theta_{h(u)} - \sum_{t=ES_{h(u)}}^{LS_{h(u)}+p'_{h(u)}} b_t^u \delta_{tm_{h(u)}} \leq 0 \qquad \forall u \in U \tag{6.21}$$

$$\gamma_i \geq 0 \qquad i = 1,..,n \tag{6.22}$$

$$\theta_h \geq 0 \qquad h = 1,..,H \tag{6.23}$$

$$\delta_{tr} \geq 0 \qquad t = 1,..,T; r = 1,..,m \tag{6.24}$$

The previous problem is equivalent to the dual problem of (aMP) with a modification in Constraint (6.21). Indeed, $t \in [ES_{h(u)}, LS_{h(u)} + p'_{h(u)}]$ instead of $t \in [1, T]$ in the dual problem.

6.3.2.3 Pricing problem

We are then searching for:

- a schedule s such that:

$$\gamma^*_{J_i(s)} - \sum_{t=1}^{T} \sum_{j=1}^{n_{J_i(s)}} a^s_{tj} \delta^*_{tmr_{ij}} > C_s \tag{6.25}$$

- or a schedule u such that:

$$\theta^*_{h(u)} - \sum_{t=ES_{h(u)}}^{LS_{h(u)}+p'_{h(u)}} b^u_t \delta^*_{tm_{h(u)}} > 0 \tag{6.26}$$

Note that expression [6.25] is the same as [6.9]. If we adopt the same reasoning as for [6.9], here also if operation o of job J_i ends at period t in a schedule s, then the contribution of the operation at period t is

$$w_{tj} = \sum_{l=t-p_{ij}+1}^{t} \delta^*_{lmr_{ij}}, \text{if } j < n_i$$

$$w_{tj} = t + \sum_{l=t-p_{ij}+1}^{t} \delta^*_{lmr_{ij}}, \text{if } j = n_i$$

Then, using the previous notation, [6.25] is equivalent to

$$\sum_{t=1}^{T} \sum_{j=1}^{n_{J_i(s)}} c^s_{tj} w_{tj} < \gamma^*_{J_i(s)} \tag{6.27}$$

where c^s_{tj} is a boolean parameter which is equal to 1 if operation O_{ij} $(1 \leq j \leq n_i)$ of job J_i ends at period t.

For every job J_i, we are searching for a schedule s verifying [6.27]. Indeed, if there is no such schedule, then no new column needs to be added.

Remarks

- We extend the pretreatment phase of Section 6.2.2.3 to discard some of the periods at which operation O_{ij} cannot be completed. In this extension the pretreatment phase taking into account the job routing constraints is still valid. Only the pretreatment taking into account the constraints from machine unavailability periods is modified as follows:
  If unavailability period h_{rk} starts in $[ES_{rk}, LS_{rk}]$, it end in $[T_{rk} - (S_{rk} - ES_{rk}), T_{rk} + (LS_{rk} - S_{rk})]$. Then all the schedules associated to h_{rk} overlap in the interval $[LS_{rk}, T_{rk} - (S_{rk} - ES_{rk})]$.
  Operation O_{ij} cannot be processed (hence finish) during the period $[S_{rk}, T_{rk}]$ reserved to an unavailability period h_{rk} on machine $r = mr_{ij}$. Then,

  $$w_{tj} = +\infty, \forall t = LS_{rk}, .., T_{rk} - (S_{rk} - ES_{rk})$$

  Note that when $LS_{rk} - ES_{rk} > T_{rk} - S_{rk}$, the length of the interval of schedules overlapping is equal to 0.

- The job sequences are found by a dynamic programming algorithm very similar to Algorithm 1 (Section 6.2.3) except that the pretreatment phase is adapted as in the previous remark or eliminated.

6.3.3 Adapted column generation algorithm

This algorithm is quite similar to Algorithm 2 (Section 6.2.3). Indeed there are some differences relative to the initial solution and adding columns. Here also one of the heuristics presented in Chapter 5 is used to calculate an initial feasible solution to the problem. It consists of one schedule or more. From the job and unavailability sequences associated to this solution, the restricted master problem (aRMP) is constructed. Then, iteratively, the linear relaxation of the (aRMP) is solved to find the dual values that are necessary to define the sequences (columns) that improve the objective value of the primal problem. These sequences are determined, by the dynamic programming algorithm for jobs and the sequence calculating procedure for unavailability periods, and added to (aRMP). The process stops when no new column is added. If the corresponding solution is integral, an optimal solution to (aRMP) is obtained, otherwise a branch-and-bound procedure is performed.

The general description of the column generation algorithm is provided by Algorithm 13.

Algorithm 13 Algorithm of the adapted column generation approach

Begin
Compute an initial feasible solution s and u to the problem.
Construct the adapted Restricted Master Program (aRMP) with the sequences of s and u
repeat
Solve the linear relaxation of (aRMP) to obtain the vector of dual values
Use a heuristic to determine if there is a sequence (or sequences) which satisfies [6.27]
if the heuristic does not find any sequence
Use the dynamic programming Algorithm 1 to determine if there is a sequence (or sequences) which satisfies [6.27]
if at least such a sequence exists
add it to (aRMP)
Search for a sequence (or sequences) for unavailability periods which satisfies [6.26]
if at least such a sequence exists
add it to (aRMP)
until no sequence is found
if the solution of (aRMP) is integral
Stop
else Use the branch-and-bound procedure to solve (aRMP)
end

6.4 Conclusion

In this chapter, a column generation approach is presented to solve the job-shop scheduling problem with and without fixed or flexible unavailability periods. Although this approach does not provide better results than the disjunctive formulation of the problem, it provides better linear relaxations in a relatively small computational time. A branch-and-bound procedure is used to solve the model with the newly added columns, but the quality of the solution quickly worsens when the problem size increases. It seems now relevant to develop a branch-and-price procedure, i.e. a procedure where new columns will be added when necessary at each node of the branch-and-bound procedure. Substantial work is however required to develop such a procedure, in particular in defining a new adapted branching scheme.

Chapter 7

Conclusions générales - General conclusions

Dans le chapitre 1, nous avons abordé tout d'abord l'ordonnancement de manière générale et de manière plus particulière l'ordonnancement de production ; et ce pour situer à la fin le contexte de notre étude. Pour cela, nous avons défini les indisponibilités des ressources et leurs flexibilités, les données, contraintes, objectifs et complexité du problème étudié.

Noua avons ensuite présenté au Chapitre 2, les deux catégories de techniques de résolution des problèmes d'optimisation combinatoire en général, et des problèmes d'ordonnancement en particulier. Pour ces deux catégories exact et approchée, les techniques les plus représentatives y ont été décrites ; cependant, seules les approches que nous avons utilisé pour résoudre notre problématique y ont été détaillées. Pour la plupart de ces méthodes, des références y ont été données pour illustrer les problèmes d'ordonnancement d'atelier avec contraintes de disponibilité des ressources.

Pour avoir une vision de l'état de la recherche pour les problèmes d'ateliers avec indisponibilité des ressources et dégager les approches intéressantes à étendre et les pistes de recherche prometteuses, nous avons discuté, dans le Chapitre 3, l'état de l'art couvrant les problèmes d'ordonnancement de production avec limitation de disponibilité des ressources à la fois avec périodes d'indisponibilité fixes et flexibles. Nous pouvons remarquer que les problèmes les plus étudiés ont été les problèmes à une machine, machines parallèles et flow shop. Les études ont essentiellement concerné les problèmes avec périodes d'indisponibilité fixes.

La première approche que nous avons utilisé est la modélisation mathématique du problème du job shop avec contraintes de disponibilité des ressources. Deux modèles mathématiques ont été d'abord présentés lorsque la préemption est non autorisée. La formulation disjonctive permet d'obtenir de meilleurs résultats que la formulation indicée par le temps, même si l'on améliore les résultats de cette dernière des données appropriées. La flexibilité sur les dates de

début des périodes d'indisponibilités.

Une formulation disjonctive générale a été ensuite proposée pour modéliser la préemption. Cette dernière ainsi que la flexibilité sur les durées des périodes d'indisponibilités se sont révélées pertinentes. Pour des problèmes de grande taille uniquemenent des solutions réalisables ont été obtenues. Ceci étant due à la forte complexité des problèmes étudiés.

Mise-à-part la résolution des jeux de données dont nous disposons, le but de cette modélisation mathématique est de permettre une meilleure connaissance des problèmes à travers leurs expressions mathématiques. Elle nous renseigne sur la façon de traiter au mieux les contraintes d'indisponibilité des ressources ; et de l'utilité de l'intégration de la flexibilité aux problèmes ; pas uniquement pour représenter la réalité de l'industrie mais aussi pour obtenir de meilleures solutions aux problèmes. Cette modélisation permet aussi d'évaluer la qualité des méthodes approchées ; et ce compte tenu qu'il est difficile de trouver de bonnes bornes théoriques aux problèmes étudiés. Elle peut aussi être facilement étendue pour considérer d'autres critères d'optimisation et inclure d'autres contraintes sur les tâches. C'est une approche souvent négligée compte tenu de la complexité des problèmes concernés ; bien qu'elle peut fournir de meilleurs résultats pour un nombre représentatif d'instances de problèmes.

Le modèle disjonctif général a été facilement étendu pour considérer d'autres critères d'optimisation et inclure d'autres contraintes sur les tâches et modéliser le problème du job shop flexible avec limitations des disponibilité des ressources.

Nous présentons dans le Chapitre 5 pour résoudre le problème du job shop avec limitation des ressources. L'introduction de la flexibilité sur les dates de début des périodes d'indisponibilité et la préemption des operations est étudiée.

Les heuristiques de construction que nous développons construisent un ordonnanacement basé sur des strétegies de décision. Le choix de ces stratégies est lié à la façon dont les priorités sur les opérations des jobs et/ou les machines, et la manière dont sont gérés les conflits entre les opérations des jobs et les périodes d'indisponibilité.

Deux sortes de méthodes sont suggérées : les heuristiques basées sur les jobs et celles basées sur les machines. Pour le premier type de méhodes, trois heuristiques dont la priorité est sur les opérations des jobs sont développées. Dans les deux premières, Job priority Heuristic (JpH) et Operation priority Heuristic 1 (OpH1), l'aspect aléatoire est introduit respectivement dans l'ordre d'insertion des jobs et des opérations. Dans la troisième heuristique OpH2, l'opération à insérer est sélectionnée par rapport à une règle donnée. Deux heuristiques basées sur les machines sont proposées. Le principe de ces méthodes consistent en associant les priorités aux machines. C'est le choix de la machine qui définit l'ensemble des opérations prêtes à être ordonnancer à sélectinner. Pour l'heuristique MOpH1, étant donné une séquence initiale d'opérations,

une fois la ready machine est définie, à partir de l'ensemble des opérations prête à être ordonnancées à exécuter sur cette machine, la première dans la séquence initiale est sélectionnée. Concernant MOpH2, elle est la combinaison de OpH2 et MOpH1. De même que pour MOpH1, la ready machine doit être définie au début de chaque itération. Ainsi, comme pour OpH2, l'opération correspondant à la règle choisie est sélectionnée à partir de l'ensemble d'opération prête à être ordonnancées à exécuter sur la ready machine.

Dans la partie expériementale, les points suivants sont discutés : nombre d'itérations, dominance des rèles de priorité, temps CPU times, préemption des opérations, flexibilité des périodes d'indisponibilité, positions initiales des périodes d'indisponibilité, priorité entre préemption et flexibilité, dominance entre heuristiques, performance des heuristiques en comparaison avec le modèle MIP.

Nous discutons ensuite la façon dont ces heuristiques de construction sont utilisées dans des méthodes afin d'améliorer les résultats du problème étudié. JoH, OpH1, MOpH1 peuvent être utilisées en tant que des blocs d'évaluation dans des méthodes qui utilisent plusieurs solutions en même temps tels que les algorithmes génétiques. OpH1 ainsi que MOpH1 peuvent être réutilisées dans une méthode qui, une fois une opération est séléctionnée est ensuite insérée, réoptimise grâce à OpH1 et MOpH1 la sous-séquence formée par les opérations précédentes dans la séquence initiale.

Pour finir, dans le Chapitre 6, nous présentons deux approches basées sur la génération de colonnes pour résoudre le problème du job shop avec des périodes d'indisponibilités fixes ou flexibles. Bien que ces méthodes ne permettent pas d'obtenir des résultats meilleurs que ceux obtenus grâce aux formulations disjonctives, elles permettent d'obtenir de meilleures relaxations linéaires en un temps court. Pour améliorer la solution obtenue suite au rajout des colonnes, nous utilisons une procédure branch-and-bound (séparation et évaluation) ; nous notons que l'efficacité de cette dernière décroit avec l'augmentation de la taille des problèmes. Pour que l'approche soit une méthode exacte, il est nécessaire de remplacer la méthode branch-and-bound par une méthode branch-and-price.

Appendix A

Appendix

Tables A.1, A.3, A.18 summarize the test results for non-preemptive operations and fixed unavailability periods with respectively JpH, OpH1 and MOpH1 heuristics and different numbers of iterations 100, 1000, 10000, 100000. Tables A.2, A.4, A.19 summarize the test results for resumable operations and flexible unavailability periods with respectively JpH, OpH1 and MOpH1 heuristics and different numbers of iterations 100, 1000, 10000, 100000. Column 1 is the name of the instance; Columns 2-3, 4-5, 6-7, 8-9 represent respectively the test results of the heuristic for 100, 1000, 10000, 100000 iterations; and Columns 10-11 give the test results for the integer resolution of the disjunctive model. Columns 2, 4, 6, 8, 10 correspond to the makespan value of the solution and Columns 3, 5, 7, 9, 11 correspond to the value of the sum of the completion dates of the jobs of the solution. To appreciate the quality of the heuristics, the best and the worst values of the objective criteria are highlighted, indications on the gap between the best solution of the heuristics and the disjunctive MIP solution are given.

Tables A.5, A.6 and A.20 present the CPU times respectively for JpH, OpH1, MOpH1 heuristics. These results are associated to the previous tables. Column 1 is the name of the instance; Columns from 2 to 5 represent the CPU times for non-preemptive operations and fixed unavailability periods; Columns from 6 to 9 represent the CPU times for resumable operations and flexible unavailability periods; and Columns 10-11 give the computation time for the integer resolution of the disjunctive model. Each Column from 2 to 9 represent the CPU of both C_{max} and $\sum C_i$ as they are calculated at the same time. Columns 2, 6 correspond to CPU for 100 iterations; Columns 3, 7 correspond to CPU for 1000 iterations; Columns 4, 8 correspond to CPU for 10000 iterations; Columns 5, 9 correspond to CPU for 100000 iterations; Columns 10 and 11 represent respectively the CPU time for C_{max}, $\sum C_i$ for the disjunctive MIP resolution. We give CPU times for each instance because the gaps between the CPU times of some instances by the disjunctive MIP are so high that it is less interesting to consider averages of the CPU times.

Tables from A.7 to A.10, (resp. from A.11 to A.14) summarize the test results for non-preemptive, resumable, non-resumable and semi-resumable operations and fixed (resp. flexible) unavailability periods with OpH2. As well, Tables from A.21 to A.24, (resp. from A.25 to A.28) presents the test results for non-preemptive, resumable, non-resumable and semi-resumable operations and fixed (resp. flexible) unavailability periods with MOpH2. Column 1 is the name of the instance; Columns 2-3, 4-5, 6-7, 8-9, 10-11, 12-13 represent respectively the tests results of the heuristic for Rules from (1) to (6); and Columns 14-15 give the test results for the integer resolution of the disjunctive model. Columns 2, 4, 6, 8, 10, 12, 14 correspond to the makespan value of the solution and Columns 3, 5, 7, 9, 11, 13, 15 correspond to the value of the sum of the completion dates of the jobs of the solution.

To show the impact of the initial position of the unavailability periods, Tables A.15 to A.16 gather respectively the test results for resumable operations in case where the flexible are initially placed in the end and the beginning of their time windows with OpH2 heuristic and different rules.

Table A.17 summarizes the test results for resumable operations and flexible unavailability periods for order CAB in OIp procedure with OpH2 heuristic. The structure of the table is the same as Table A.12.

A.1 Job based heuristics

Table A.1: Test results for non-preemptive operations and fixed unavailability periods with JpH heuristic and different numbers of iterations.

Problem	100 iterations		1000 iterations		10000 iterations		100000 iterations		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	**961**∼	**3748**∼	**961**∼	**3748**∼	**961**∼	**3748**∼	**961**∼	**3748**∼	895*	3652*
5m5n2	**1112**∼	**4669***	**1112**∼	**4669***	**1112**∼	**4669***	**1112**∼	**4669***	1096*	4669*
5m5n3	*1194*	**4008***	**1157**∼	**4008***	**1157**∼	**4008***	**1157**∼	**4008***	1070*	4008*
5m5n4	**1147***	**4584***	**1147***	**4584***	**1147***	**4584***	**1147***	**4584***	1147*	4584*
5m5n5	**1202***	*4683*	**1202***	**4663***	**1202***	**4663***	**1202***	**4663***	1202*	4663*
5m10n1	*1630*	*11354*	1590	10891	1530	10561	**1471**∼	**10304**∼	1361*	9850*
5m10n2	*1731*	*11756*	**1521**∼	11404	**1521**∼	11267	**1521**∼	**11036**∼	1455	10637*
5m10n3	*1777*	*12130*	1754	12087	**1722**∼	11670	**1722**∼	**11594**∼	1607	11072
5m10n4	*1767*	*12159*	1667	11788	1658	11534	**1619**∼	**11353**∼	1537*	10630*
5m10n5	*1590*	*11173*	1555	10816	1535	10281	**1515**∼	**10179**∼	1415	9706*
10m10n1	*2431*	*18350*	2421	18089	**2226**	**17308**∼	2231	17341	1948	16878
10m10n2	*2334*	*17866*	2252	17430	2241	16543	**2202**	**16539**∼	1917*	15777*
10m10n3	*2238*	*17171*	2158	16739	**2098**	16864	**2098**	**16238**∼	1875	15352*
10m10n4	2110	16236	*2129*	*16855*	**1990**	16147	**1990**	**16143**∼	1772*	15374*
10m10n5	*2365*	*17452*	2151	16861	2135	**16132**∼	**2105**∼	16182	1975	15625*
10m15n1	*2749*	*30054*	2739	29088	**2550**	29125	2594	**28661**∼	2089	27199
10m15n2	*3032*	*32484*	2873	31784	2762	30908	**2742**	**30743**∼	2316	29186
10m15n3	*2901*	*30955*	2713	29816	2668	29390	**2513**	**28814**+	2261	u
10m15n4	*2866*	*30836*	2724	30008	2622	29378	**2532**	**28436**∼	2254	27423
10m15n5	*2782*	*31799*	2802	31371	2725	30377	**2662**	**30114**∼	2282	28609

bold: best solution

italic: worst solution

*: optimal solution

u: unknown solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.2: Test results for resumable operations and flexible unavailability periods with JpH heuristic and different numbers of iterations.

Problem	100 iterations		1000 iterations		10000 iterations		100000 iterations		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	*907*	**3597**∼	**825***	**3597**∼	**825***	**3597**∼	**825***	**3597**∼	825*	3356*
5m5n2	**1076**∼	**4468**∼	**1076**∼	**4468**∼	**1076**∼	**4468**∼	**1076**∼	**4468**∼	977*	4174*
5m5n3	*1147*	**3968**∼	**1137**	**3968**∼	**1137**	**3968**∼	**1137**	**3968**∼	1020*	3892*
5m5n4	**1108***	**4242**∼	**1108***	**4242**∼	**1108***	**4242**∼	**1108***	**4242**∼	1108*	4136*
5m5n5	**1190**∼	**4646**∼	**1190**∼	**4646**∼	**1190**∼	**4646**∼	**1190**∼	**4646**∼	1108*	4417*
5m10n1	*1593*	*11068*	1500	10490	1493	10373	**1440**∼	**10119**∼	1316	9484
5m10n2	*1644*	*11294*	1501	10937	1501	10826	**1464**∼	**10778**∼	1400	10364
5m10n3	*1828*	11899	1757	*11938*	1709	**11307**∼	**1702**∼	11315	1563	10710
5m10n4	*1739*	*11737*	1698	11596	**1618**∼	11194	**1618**∼	**10975**∼	1492	10439
5m10n5	*1600*	*10773*	1529	10101	1506	10072	**1486**∼	**9860**∼	1372	9101
10m10n1	*2390*	17787	2291	*17794*	2232	17409	**2202**	**17348**∼	1912	16397
10m10n2	*2381*	*17453*	2255	16680	2192	16572	**2136**	**16448**∼	1915	15190
10m10n3	*2191*	*16901*	2118	16661	2101	16117	**2031**	**15829**∼	1726	15261
10m10n4	*2110*	*16619*	2031	16181	2060	15853	**1970**	**15772**∼	1701	14876
10m10n5	*2255*	*16827*	2055	16534	2071	16197	**2032**∼	**15697**∼	1839	15273
10m15n1	*2801*	*29002*	2737	28996	2621	28629	**2549**∼	**28192**∼	2087	26762
10m15n2	*3012*	*31308*	2854	31007	2743	30364	**2596**∼	**30074**∼	2450	28685
10m15n3	*2762*	*30257*	2629	29338	2611	28760	**2517**+	**28619**∼	4202	27221
10m15n4	*2840*	*29968*	2664	28914	2574	28704	**2552**+	**28379**∼	4941	27264
10m15n5	*2869*	*31202*	2765	29660	2692	29794	**2552**∼	**29486**∼	2292	28000

bold: best solution

italic: worst solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.3: Test results for non-preemptive operations and fixed unavailability periods with OpH1 heuristic and different numbers of iterations.

Problem	100 iterations		1000 iterations		10000 iterations		100000 iterations		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	**895***	*3756*	**895***	3671	**895***	**3652***	**895***	**3652***	895*	3652*
5m5n2	**1096***	*4706*	**1096***	*4706*	**1096***	**4669***	**1096***	**4669***	1096*	4669*
5m5n3	**1070***	*4471*	**1070***	4188	**1070***	4130	**1070***	**4008***	1070*	4008*
5m5n4	*1235*	*4789*	**1147***	**4584***	**1147***	**4584***	**1147***	**4584***	1147*	4584*
5m5n5	**1202***	*4831*	**1202***	4725	**1202***	**4663***	**1202***	**4663***	1202*	4663*
5m10n1	*1460*	*11574*	**1420**∼	11288	**1420**∼	10802	**1420**∼	**10759**∼	1361*	9850*
5m10n2	*1545*	*12311*	*1545*	12185	1525	12252	**1455**∼	**11775**	1455	10637*
5m10n3	*1677*	*13382*	1652	12455	1637	12449	**1607**∼	**12234**∼	1607	11072
5m10n4	*1678*	*12325*	1638	12249	1607	12065	**1557**∼	**11813**	1537*	10630*
5m10n5	*1520*	*11696*	1510	11315	1495	11109	**1425**∼	**10410**∼	1415	9706*
10m10n1	*2181*	*19074*	2117	18731	2087	18136	**2030**	**17935**∼	1948	16878
10m10n2	*2214*	*19032*	2112	18378	2052	17544	**2022**∼	**17342**∼	1917*	15777*
10m10n3	*2065*	*18064*	1950	17166	1935	17014	**1912**∼	**16641**∼	1875	15352*
10m10n4	*2035*	*17831*	1970	17369	1931	16783	**1860**∼	**16331**∼	1772*	15374*
10m10n5	*2115*	*17893*	2047	17438	1985	16812	**1975**∼	**16795**∼	1975	15625*
10m15n1	*2479*	*32252*	2399	30658	**2320**	30908	2333	**30071**∼	2089	27199
10m15n2	*2600*	*33784*	*2600*	33515	**2490**∼	**32194**∼	**2490**∼	32357	2316	29186
10m15n3	*2533*	*33397*	2503	32425	2473	32105	**2423**∼	**31213**+	2261	u
10m15n4	2462	*32320*	*2492*	31846	2338	31167	**2316**∼	**30781**	2254	27423
10m15n5	*2562*	*33716*	2538	32693	2482	32639	**2462**∼	**31787**	2282	28609

bold: best solution

italic: worst solution

*: optimal solution

u: unknown solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.4: Test results for resumable operations and flexible unavailability periods with OpH1 heuristic and different numbers of iterations.

Problem	100 iterations		1000 iterations		10000 iterations		100000 iterations		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	**825***	*3668*	**825***	3576	**825***	**3552**∼	**825***	**3552**∼	825*	3356*
5m5n2	**1076**∼	*4537*	**1076**∼	4476	**1076**∼	**4412**∼	**1076**∼	**4412**∼	977*	4174*
5m5n3	**1034**∼	*4187*	**1034**∼	4088	**1034**∼	4012	**1034**∼	**3968**∼	1020*	3892*
5m5n4	*1174*	*4581*	1127	4332	**1108***	**4242**∼	**1108***	**4242**∼	1108*	4136*
5m5n5	**1182**	4732	**1182**	4632	**1182**∼	**4565**∼	**1182**∼	4569	1108*	4417*
5m10n1	*1457*	*11292*	1437	11088	1390	10714	**1360**∼	**10448**∼	1316	9484
5m10n2	*1501*	*12055*	1475	12026	1430	11544	**1404**∼	**11388**∼	1400	10364
5m10n3	*1652*	12537	1632	*12667*	1617	12207	**1587**∼	**11805**∼	1563	10710
5m10n4	*1668*	*12489*	1578	12108	1566	11805	**1518**∼	**11327**∼	1492	10439
5m10n5	*1535*	*11849*	1475	11411	1430	10633	**1405**∼	**10533**	1372	9101
10m10n1	*2110*	17714	2054	*18207*	1994	18049	**1991**∼	**17614**∼	1912	16397
10m10n2	*2102*	*18405*	2087	17954	2024	17273	**1982**∼	**16855**∼	1915	15190
10m10n3	*2032*	*17080*	1930	16763	1890	16354	**1836**∼	**16205**∼	1726	15261
10m10n4	1950	*17481*	*1959*	16616	1900	16501	**1847**∼	**16033**∼	1701	14876
10m10n5	*2003*	*17045*	1946	16949	1955	16456	**1913**∼	**16153**∼	1839	15273
10m15n1	2341	*31295*	*2359*	30384	2301	29907	**2282**∼	**29521**∼	2087	26762
10m15n2	*2600*	*33940*	2570	32598	2480	**31752**	**2440**+	31860	2450	28685
10m15n3	2443	*32797*	*2460*	31005	2421	31135	**2277**+	**30292**	4202	27221
10m15n4	*2424*	*31814*	2412	30851	2374	30624	**2342**+	**30095**∼	4941	27264
10m15n5	*2518*	*33002*	2488	32343	2455	31391	**2376**∼	**31389**	2292	28000

bold: best solution

italic: worst solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.5: CPU time for JpH heuristic and different numbers of iterations.

Problem	Non-preemptive operations and fixed unavailability periods						Resumable operations and flexible unavailability periods					
	JpH				MIP		JpH				MIP	
	100 iterations	1000 iterations	10000 iterations	100000 iterations	C_{max}	$\sum C_i$	100 iterations	1000 iterations	10000 iterations	100000 iterations	C_{max}	$\sum C_i$
5m5n1	0.01	0.04	0.29	2.73	0.09	0.04	0	0.01	0.34	3.05	3.55	0.90
5m5n2	0	0.04	0.26	2.69	0.02	0.01	0	0.03	0.29	2.99	1.23	0.12
5m5n3	0	0.03	0.28	2.71	0.23	0.04	0	0.03	0.29	3.04	1.07	1.98
5m5n4	0	0.03	0.28	2.69	0.27	0.06	0	0.03	0.31	3.01	0.89	1.04
5m5n5	0.01	0.01	0.26	2.69	0.51	0.07	0.01	0.03	0.29	2.99	10.82	2.37
5m10n1	0	0.04	0.49	4.86	43.09	672.04	0.01	0.06	0.57	5.67	3496.37	3600
5m10n2	0.01	0.04	0.48	4.92	1519.4	167.89	0	0.04	0.56	5.64	3449.11	3600
5m10n3	0	0.04	0.49	4.93	1939.35	3600	0.01	0.06	0.56	5.70	3489.86	3600
5m10n4	0	0.06	0.49	4.94	1324.87	565	0.01	0.06	0.57	5.64	3498.93	3600
5m10n5	0	0.04	0.51	4.91	174.72	1566.81	0	0.06	0.57	5.64	3451.59	3600
10m10n1	0.01	0.09	0.92	9.17	2073.96	7.40	0.01	0.10	1.07	10.68	3561.88	3600
10m10n2	0	0.09	0.90	9.20	664.17	547.82	0.01	0.10	1.06	10.71	3535.8	3600
10m10n3	0	0.09	0.92	9.11	466.46	646.60	0.01	0.10	1.06	10.57	3552.65	3600
10m10n4	0.01	0.09	0.92	9.25	669.87	3600	0.01	0.10	1.06	10.71	3552.26	3600
10m10n5	0.01	0.09	0.93	9.15	1814.3	1237.39	0.01	0.11	1.07	10.71	3535.78	3600
10m15n1	0.01	0.17	1.74	17.62	2708.88	3600	0.03	0.20	2.02	20.34	3573.83	3600
10m15n2	0.01	0.18	1.76	17.58	1893.73	3600	0.01	0.20	2.02	20.65	3532.5	3600
10m15n3	0.01	0.17	1.74	17.61	2645.88	3600	0.03	0.20	2.04	20.32	3555.99	3600
10m15n4	0.01	0.17	1.76	17.59	2679.33	3600	0.01	0.21	2.02	20.29	3542.84	3600
10m15n5	0.01	0.17	1.74	17.59	2807.98	3600	0.01	0.20	2.02	20.33	3568.95	3600

Table A.6: CPU time for OpH1 heuristic and different numbers of iterations.

Problem	Non-preemptive operations and fixed unavailability periods						Resumable operations and flexible unavailability periods					
	OpH1				MIP		OpH1				MIP	
	100 iterations	1000 iterations	10000 iterations	100000 iterations	C_{max}	$\sum C_i$	100 iterations	1000 iterations	10000 iterations	100000 iterations	C_{max}	$\sum C_i$
5m5n1	0	0.03	0.32	2.90	0.09	0.04	0	0.04	0.31	3.21	3.55	0.90
5m5n2	0	0.03	0.29	2.85	0.02	0.01	0	0.04	0.31	3.15	1.23	0.12
5m5n3	0.01	0.03	0.28	2.87	0.23	0.04	0.01	0.03	0.31	3.19	1.07	1.98
5m5n4	0	0.03	0.29	2.87	0.27	0.06	0	0.03	0.32	3.16	0.89	1.04
5m5n5	0.01	0.01	0.29	2.85	0.51	0.07	0	0.03	0.31	3.19	10.82	2.37
5m10n1	0	0.04	0.51	5.19	43.09	672.04	0.01	0.06	0.59	5.95	3496.37	3600
5m10n2	0.01	0.04	0.53	5.27	1519.4	167.89	0	0.06	0.59	5.95	3449.11	3600
5m10n3	0	0.04	0.53	5.25	1939.35	3600	0.01	0.06	0.60	5.97	3489.86	3600
5m10n4	0	0.06	0.51	5.22	1324.87	565	0	0.06	0.57	5.80	3498.93	3600
5m10n5	0.01	0.04	0.51	5.11	174.72	1566.81	0.01	0.06	0.59	5.83	3451.59	3600
10m10n1	0.01	0.10	1.06	10.32	2073.96	7.40	0.01	0.12	1.18	11.91	3561.88	3600
10m10n2	0	0.10	1.04	10.53	664.17	547.82	0.01	0.12	1.20	12.13	3535.8	3600
10m10n3	0	0.10	1.03	10.35	466.46	646.60	0.01	0.10	1.17	11.87	3552.65	3600
10m10n4	0.01	0.10	1.06	10.57	669.87	3600	0.01	0.10	1.21	12.16	3552.26	3600
10m10n5	0.01	0.11	1.04	10.42	1814.3	1237.39	0.01	0.10	1.20	12.01	3535.78	3600
10m15n1	0.01	0.18	1.96	19.65	2708.88	3600	0.01	0.23	2.24	22.35	3573.83	3600
10m15n2	0.01	0.18	1.95	19.56	1893.73	3600	0.03	0.21	2.21	22.19	3532.5	3600
10m15n3	0.01	0.20	1.96	19.70	2645.88	3600	0.03	0.21	2.23	22.38	3555.99	3600
10m15n4	0.03	0.21	1.96	19.57	2679.33	3600	0.01	0.21	2.21	22.15	3542.84	3600
10m15n5	0.01	0.18	1.95	19.65	2807.98	3600	0.03	0.21	2.23	22.23	3568.95	3600

Table A.7: Test results for non-preemptive operations and fixed unavailability periods with OpH2 heuristic and different rules.

Problem	Rule 1		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	914	**4094**	*1318*	4304	1015	*4551*	914	4152	**895***	4310	**895***	4310	895*	3652*
5m5n2	1112	4846	*1336*	*5671*	1213	**4823**∼	1213	5067	1143	5317	**1096***	5077	1096*	4669*
5m5n3	1134	4929	*1430*	**4735**	1132	4961	**1122**∼	4857	1222	*5293*	1192	4813	1070*	4008*
5m5n4	**1237**∼	5261	*1389*	**4890**∼	1379	5691	1255	5375	**1237**∼	5261	1387	*5855*	1147*	4584*
5m5n5	1332	*5659*	*1369*	5557	**1290**∼	5329	1339	**5229**	1332	*5659*	1332	5499	1202*	4663*
5m10n1	1700	*14414*	*1747*	**11312**	1720	13399	1580	11531	**1560**	13053	1593	12918	1361*	9850*
5m10n2	1635	**12897**	*1815*	13519	1645	14157	1805	13539	1635	*14302*	**1495**∼	13612	1455	10637*
5m10n3	1897	16077	1884	**12639**	*1937*	15785	1917	15295	1897	*16425*	**1877**	16019	1607	11072
5m10n4	1788	14884	*1947*	**11839**	1868	13704	1797	14399	1788	*15035*	**1767**	14864	1537*	10630*
5m10n5	1625	13530	1624	**11473**	**1615**	12765	1645	12336	*1685*	*14212*	**1615**	13290	1415	9706*
10m10n1	**2121**∼	19685	2357	**18412**∼	2221	20860	2300	20271	2163	20304	*2437*	*21225*	1948	16878
10m10n2	**2214**	20120	2429	19585	2399	*20556*	*2639*	**19418**	**2214**	20120	2452	20240	1917*	15777*
10m10n3	2072	**18538**	2285	19003	*2615*	*20072*	2241	19123	**2062**∼	19704	2175	19825	1875	15352*
10m10n4	**1990**	**18004**	2307	18138	*2386*	19264	2320	18413	2040	18356	2320	*19276*	1772*	15374*
10m10n5	**2115**∼	19379	2303	**17163**∼	2222	18324	2251	18164	2132	*19757*	*2312*	19125	1975	15625*
10m15n1	**2420**∼	32695	2749	**29663**	*2899*	34255	2819	34711	2549	*35566*	2530	33453	2089	27199
10m15n2	2615	35782	2862	**33848**	*2980*	*36405*	2870	35328	**2585**	35535	2677	35205	2316	29186
10m15n3	**2621**	33712	2728	**31308**+	2972	35703	*3007*	35051	2663	35121	2763	*36641*	2261	u
10m15n4	2494	31401	*2802*	**30935**	2776	34327	2724	*35478*	**2452**∼	33028	2546	33309	2254	27423
10m15n5	**2445**∼	33758	2784	**30911**∼	*3025*	*36304*	2732	34402	2572	35277	2632	35252	2282	28609

bold: best solution

italic: worst solution

*: optimal solution

u: unknown solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.8: Test results for resumable operations and fixed unavailability periods with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**895***	**3959**	1178	3969	1045	4070	935	4194	**895***	4117	**895***	4117	845*	3577*
5m5n2	1080	**4482**	1320	5274	1158	4644	1096	4607	**997***	4530	1000	4670	997*	4385*
5m5n3	1126	4610	1240	**4570**	**1090**	4672	**1090**	4672	1126	5438	1100	4618	1020*	3912*
5m5n4	1228	5158	1268	**4136***	1255	5325	1255	5365	**1177**	5000	1398	5910	1135*	4136*
5m5n5	1168	5014	1210	**4986**	1316	5066	1282	5232	1332	5924	1266	4992	1108*	4443*
5m10n1	1560	12572	1633	**10600**	1720	13119	1720	12122	**1533**	13267	1560	13363	1322	9610
5m10n2	**1460**	12322	1658	12650	1685	**12296**	1671	13533	1551	13790	1524	13324	1400	10402
5m10n3	**1740**	14966	1838	**12417**	2007	15463	1959	15495	1834	15736	1834	15544	1583	11130
5m10n4	**1617**	13281	1842	**11563**	1906	13734	1928	14795	1771	14601	1638	13346	1492	10526
5m10n5	**1486**	12209	1697	**10870**	1746	12542	1715	12375	1601	13267	**1486**	12173	1386	9407
10m10n1	**2021**	19047	2263	**18529**	2648	19860	2254	19740	2102	19701	2239	19571	1907	16335
10m10n2	**2218**	19967	2392	**18373**	2642	22154	2536	21495	2228	20264	2632	20657	1822	15353
10m10n3	**1943**	17589	2105	**17200**	2290	19288	2325	18917	2078	19315	2093	18840	1815	15349
10m10n4	**1898**	17366	2004	**16577**	2341	20393	2130	18675	2091	18723	2295	18944	1755	15423
10m10n5	**1977**	17414	2047	**16648**	2275	19220	2247	18122	2043	18406	2004	17695	1870	15669
10m15n1	**2377**	31579	2468	**28922**	3043	33400	2760	33000	2476	34934	2528	33146	2310	27010
10m15n2	**2466**	33748	2784	**33268**	3395	36271	2970	36866	2572	34776	2677	35172	2460	29251
10m15n3	**2511**	31953	2721	**30690**	2907	32620	2857	33975	2616	34816	2660	33684	2313	27196
10m15n4	**2441**	31767	2594	**29943**	2836	33151	2830	33221	2543	33509	2527	32805	6958	27530
10m15n5	**2406**	33225	2662	**30212**	2902	34807	3045	35946	2442	33615	2802	35288	6392	27601

bold: best solution

*: optimal solution

Table A.9: Test results for non-resumable operations and fixed unavailability periods with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	914	**4094**	1318	4304	1045	4144	935	4257	**895***	4310	**895***	4310	895*	3652*
5m5n2	1112	**4706**	1336	5671	1163	4713	1112	**4706**	1143	5317	**1096***	5077	1096*	4669*
5m5n3	1134	4929	1430	**4735**	**1122**	4799	**1122**	4799	1272	5806	1192	4813	1070*	4008*
5m5n4	1385	5827	1389	**4890**	1255	5325	1255	5365	**1237**	5261	1529	6137	1147*	4584*
5m5n5	1332	5747	1369	5557	**1280**	**5064**	1390	5645	1332	5797	1412	5659	1202*	4663*
5m10n1	1700	14491	1747	**11312**	1720	13372	1780	12609	**1560**	13384	1593	13249	1361	9850
5m10n2	1635	13596	1815	13519	1725	**12682**	1771	13889	1635	14302	**1495**	13612	1455	10643
5m10n3	1897	16077	1884	**12639**	2007	15515	1959	15526	1897	16425	**1877**	16019	1637	11072
5m10n4	1788	14884	1947	**11839**	1806	13234	**1738**	14977	1788	15035	1767	14864	1537	10630
5m10n5	1685	13908	1624	**11473**	1746	13237	**1615**	12769	1685	14212	1675	13733	1415	9706
10m10n1	**2178**	19274	2357	**18412**	2438	20910	2254	19740	2310	20993	2437	21545	1987	17117
10m10n2	**2202**	19915	2429	**19585**	2582	22325	2436	21897	2214	20120	2632	20700	1917	15777
10m10n3	**2102**	**18581**	2285	19003	2435	19990	2325	19084	2235	20915	2175	19825	1875	15644
10m10n4	2067	18358	2307	**18138**	2341	20393	2230	18911	**2040**	18356	2320	19276	1772	15374
10m10n5	**2115**	19384	2303	**17163**	2325	19554	2260	19198	2321	20562	2312	19235	1933	15719
10m15n1	**2437**	34064	2749	**29663**	3029	33218	2990	34390	2729	36853	2699	35944	2209	26886
10m15n2	**2586**	35173	2862	**33848**	3072	35694	2970	36886	2595	36107	2836	37101	2326	u
10m15n3	**2631**	34390	2728	**31308**	2893	34877	3163	36356	2663	35121	2813	36501	2213	28308
10m15n4	**2494**	31696	2802	**30935**	2936	34592	3040	34241	2692	34524	2722	35226	2212	27763
10m15n5	**2445**	33758	2784	**30911**	2928	35439	3135	35280	2682	36710	2855	37082	2342	19226.97

bold: best solution

*: optimal solution

u: unknown solution

Table A.10: Test results for semi-resumable operations and fixed unavailability periods with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**895**	4011.75	1242	**4004.5**	1045	4132	935	4223.5	**895**	4242.875	**895**	4242.87	866.62*	3613.00*
5m5n2	1096	**4604.5**	1336	5431.25	1163	4713	1096	4656.5	**1047.25**	4907.75	1059.87	4916.12	1042.25*	4601.00*
5m5n3	1133	**4640**	1370	4763.25	**1104.75**	4755.75	**1104.75**	4755.75	1217	5367	1149	4716	1046.50*	3938.00*
5m5n4	1327	5595	1338	**4310**	1255	5325	1255	5365	**1198.5**	5103	1456	6200	1137.00*	4307.00*
5m5n5	**1272**	5418	1340	5415.5	1368	**5265**	1344	5452	1332	5924	1352	5383.5	1159.50*	4597.50*
5m10n1	**1560**	13317.5	1747	**11174.5**	1720	13153	1780	12256	**1560**	13744.18	**1560**	13637.43	1378.50	9843.00
5m10n2	1535.25	12975.25	1794	13383	1685	**12246**	1713	13827	1595	14011	**1496.5**	13139.75	1452.75	10440.50
5m10n3	**1839.5**	15587.5	1861.5	**12658.5**	2007	15489	1959	15510.5	1865.5	16159.25	1855.5	15767.5	1625.50	11106.00
5m10n4	1713.75	14133.75	1843.5	**12094**	1806	13234	1812.5	15343.5	1788	14905	**1661**	13720	1503.50	10588.00
5m10n5	**1548.25**	12804.5	1686	**11260.5**	1746	12647	1615	12688.5	1748	14528.75	**1548.25**	12934	1406.00	9515.00
10m10n1	**2084.5**	18776.25	2160.5	**18593.5**	2648	20060	2254	19740	2199	20218	2331	20732	1906.00	16806.00
10m10n2	2272.5	20653.5	2326	**19325**	2582	22059.5	2536	21505	**2212**	19938.75	2632	20675	1957.50	15572.70
10m10n3	**2062**	**18300**	2415	18863	2429.5	19946	2325	18930.5	2068.5	19424.5	2148.5	19565.25	1825.25	15352.00
10m10n4	**1959.12**	17793.37	2399.75	**17774.75**	2341	20393	2130	18763	2064.25	18435.5	2295	19225	1780.00	15411.20
10m10n5	**2100.5**	18740.75	2129.5	**17190**	2275	19220	2217	18472	2115	18920.62	2305.37	19139.06	1938.37	15595.00
10m15n1	**2501.75**	33197.5	2734.5	**29446.37**	2893	33511	2760	33380	2587.75	36567	2526.25	35169.75	2107.00	26993.20
10m15n2	**2503.75**	34460	2844	**33122**	3192	35904	2970	36886	2595	35929.5	2784	37154	2366.00	u
10m15n3	**2521**	32589.25	2683	**30666.5**	2893	34877	3063	34674	2666.87	35307.62	2792.81	34994.87	2271.00	u
10m15n4	**2536.37**	32192.12	2841.75	**31041.5**	2946	33728.5	3040	33691	2544	33378	2623	32930.75	2214.00	27277.20
10m15n5	**2488**	33864	2787.75	**31149.12**	2928	35337	2909	35559	2556.37	34929.25	2778.25	35562.5	2272.25	28805.20

bold: best solution

*: optimal solution

u: unknown solution

Table A.11: Test results for non-preemptive operations and flexible unavailability periods with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**	1254	4043	939	4145	871	3925	875	4210	875	4210	825*	3552*
5m5n2	**1076***	4757	1316	5571	**1076***	4775	**1076***	**4747**	**1076***	4757	**1076***	4765	1076*	4412*
5m5n3	1114	4579	1410	**4370**	**1042**	4603	**1042**	4603	1132	5055	1100	4479	1034*	3968*
5m5n4	**1217**	5045	1369	**4335**	1359	5591	1262	5320	**1217**	5157	1398	5910	1108*	4242*
5m5n5	**1211**	5045	**1211**	5000	1252	5203	1252	**4928**	1312	5835	1392	5559	1182*	4565*
5m10n1	1580	13512	1727	**10992**	1742	13752	1890	13321	**1540**	12748	1543	12789	1300*	9535*
5m10n2	**1525**	**12624**	1795	13242	1641	13464	1925	13445	1551	13236	**1525**	13091	1400	10256*
5m10n3	**1740**	14352	1798	**12025**	1916	14471	1937	14783	1877	16225	1857	15707	1578	10997
5m10n4	**1628**	13360	1877	**11803**	1858	13814	1796	13658	1768	14835	1667	13548	1492	10491*
5m10n5	1605	12920	1904	**12082**	1832	13282	1727	13495	1605	12960	**1595**	12484	1372*	9101
10m10n1	**2011**	18487	2337	**17150**	2393	20219	2296	20147	2166	20271	2291	20262	1859	16085
10m10n2	**2147**	19352	2402	**18189**	2379	20572	2542	19944	2197	20035	2432	20040	1839*	15200*
10m10n3	**2042**	18281	2372	18441	2534	18467	2201	**18182**	**2042**	19504	2155	19494	1773*	15164
10m10n4	**1911**	16852	2217	**16239**	2180	18284	2069	17399	2020	18092	2350	18870	1690*	15155
10m10n5	2095	18099	**2005**	**16566**	2227	18269	2135	18237	2095	19256	2152	18805	1833	15136
10m15n1	**2338**	31937	2537	**30120**	2766	34472	2679	33615	2549	35767	2509	34063	2103	27221
10m15n2	**2566**	34467	2600	33996	2913	36137	2780	**33794**	2575	35807	2657	34993	2388	27977
10m15n3	**2515**	32593	2610	**30318**	2865	33441	2958	33425	2601	34711	2613	33899	2221	u
10m15n4	**2474**	31312	2691	**30082**	2972	35350	2984	34169	2562	33470	2536	32766	2120	27143
10m15n5	**2469**	33559	2841	**30419**	2872	34241	2535	33715	2512	34410	2792	35480	2225	28312

bold: best solution

*: optimal solution

u: unknown solution

Table A.12: Test results for resumable operations and flexible unavailability periods with OpH2 heuristic and different rules.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**∼	*1254*	4043	939	4145	871	3925	875	*4210*	875	*4210*	825*	3356*
5m5n2	**1076**∼	4757	*1316*	*5571*	**1076**	4775	**1076**	**4747**	**1076**	4757	**1076**	4765	977*	4174*
5m5n3	1114	4579	*1410*	**4370**	**1042**∼	4603	**1042**∼	4603	1132	*5055*	1100	4479	1020*	3892*
5m5n4	**1217**∼	5045	*1369*	**4335**∼	1359	5591	1262	5320	**1217**∼	5157	1398	*5910*	1108*	4136*
5m5n5	**1211**∼	5045	**1211**∼	5000	1252	5203	1252	**4928**	1312	*5835*	1392	5559	1108*	4417*
5m10n1	1580	13512	1727	**10992**	1742	*13752*	*1890*	13321	**1540**	12748	1543	12789	1316	9484
5m10n2	**1525**∼	**12624**	1795	13242	1641	*13464*	*1925*	13445	1551	13236	**1525**∼	13091	1400	10364
5m10n3	**1740**	14352	1798	**12025**	1916	14471	*1937*	14783	1877	*16225*	1857	15707	1563	10710
5m10n4	**1628**∼	13360	*1877*	**11803**	1858	13814	1796	13658	1768	*14835*	1667	13548	1492	10439
5m10n5	1605	12920	*1904*	**12082**	1832	13282	1727	*13495*	1605	12960	**1595**	12484	1372	9101
10m10n1	**2011**∼	18487	2337	**17150**∼	*2393*	20219	2296	20147	2166	*20271*	2291	20262	1912	16397
10m10n2	**2147**	19352	2402	**18189**	2379	*20572*	*2542*	19944	2197	20035	2432	20040	1915	15190
10m10n3	**2042**	18281	2372	18441	*2534*	18467	2201	**18182**	**2042**	*19504*	2155	19494	1726	15261
10m10n4	**1911**	16852	2217	**16239**∼	2180	18284	2069	17399	2020	18092	*2350*	*18870*	1701	14876
10m10n5	2095	18099	**2005**∼	**16566**∼	*2227*	18269	2135	18237	2095	*19256*	2152	18805	1839	15273
10m15n1	**2338**	31937	2537	**30120**	*2766*	34472	2679	33615	2549	*35767*	2509	34063	2087	26762
10m15n2	**2566**∼	34467	2600	33996	*2913*	*36137*	2780	**33794**	2575	35807	2657	34993	2450	28685
10m15n3	**2515**+	32593	2610	**30318**	2865	33441	*2958*	33425	2601	*34711*	2613	33899	4202	27221
10m15n4	**2474**+	31312	2691	**30082**∼	2972	*35350*	*2984*	34169	2562	33470	2536	32766	4941	27264
10m15n5	**2469**∼	33559	2841	**30419**∼	*2872*	34241	2535	33715	2512	34410	2792	*35480*	2292	28000

bold: best solution

italic: worst solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.13: Test results for non-resumable operations and flexible unavailability periods with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**	1254	4043	939	4145	871	3925	875	4210	875	4210	825*	3352*
5m5n2	**1076***	4757	1316	5571	**1076***	4775	**1076***	**4747**	**1076***	4757	**1076***	4765	1076*	4412*
5m5n3	1114	4579	1410	**4370**	**1042**	4603	**1042**	4603	1132	5055	1100	4479	1034*	3968*
5m5n4	**1217**	5045	1369	**4335**	1359	5591	1262	5320	**1217**	5157	1398	5910	1108*	4242*
5m5n5	**1211**	5045	1211	5000	1252	5203	1252	**4928**	1312	5835	1392	5559	1182*	4565*
5m10n1	1580	13512	1727	**10992**	1742	13752	1890	13321	**1540**	12748	1543	12789	1302	9585
5m10n2	**1525**	**12624**	1795	13242	1641	13464	1925	13445	1551	13236	**1525**	13091	1400	10274
5m10n3	**1740**	14352	1798	**12025**	1916	14471	1937	14783	1877	16225	1857	15707	1578	10558
5m10n4	**1628**	13360	1877	**11803**	1858	13814	1796	13658	1768	14835	1667	13548	1492	10493
5m10n5	1605	12920	1904	**12082**	1832	13282	1727	13495	1605	12960	**1595**	12484	1377	9292
10m10n1	**2011**	18487	2337	**17150**	2393	20219	2296	20147	2166	20271	2291	20262	1912	16297
10m10n2	**2147**	19352	2402	**18189**	2379	20572	2542	19944	2197	20035	2432	20040	1874	15213
10m10n3	**2042**	18281	2372	18441	2534	18467	2201	**18182**	**2042**	19504	2155	19494	1773	15278
10m10n4	**1911**	16852	2217	**16239**	2180	18284	2069	17399	2020	18092	2350	18870	1690	15045
10m10n5	2095	18099	**2005**	**16566**	2227	18269	2135	18237	2095	19256	2152	18805	1880	15347
10m15n1	**2338**	31937	2537	**30120**	2766	34472	2679	33615	2549	35767	2509	34063	2137	u
10m15n2	**2566**	34467	2600	33996	2913	36137	2780	**33794**	2575	35807	2657	34993	2395	28004
10m15n3	**2515**	32593	2610	**30318**	2865	33441	2958	33425	2601	34711	2613	33899	2403	27669
10m15n4	**2474**	31312	2691	**30082**	2972	35350	2984	34169	2562	33470	2536	32766	2166	28015
10m15n5	**2469**	33559	2841	**30419**	2872	34241	2535	33715	2512	34410	2792	35480	2146	27763

bold: best solution

*: optimal solution

u: unknown solution

Table A.14: Test results for semi-resumable operations and flexible unavailability periods with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**	1254	4043	939	4145	871	3925	875	4210	875	4210	825.00*	3366.50*
5m5n2	**1076**	4757	1316	5571	**1076**	4775	1076	**4747**	**1076**	4757	**1076**	4765	1000.00*	4218.25*
5m5n3	1114	4579	1410	**4370**	**1042**	4603	**1042**	4603	1132	5055	1100	4479	1031.50*	3918.00*
5m5n4	**1217**	5045	1369	**4335**	1359	5591	1262	5320	**1217**	5157	1398	5910	1108.00*	4220.50*
5m5n5	**1211**	5045	**1211**	5000	1252	5203	1252	**4928**	1312	5835	1392	5559	1145.00*	4528.00
5m10n1	1580	13512	1727	**10992**	1742	13752	1890	13321	**1540**	12748	1543	12789	1339.25	9535.00
5m10n2	**1525**	**12624**	1795	13242	1641	13464	1925	13445	1551	13236	**1525**	13091	1400.00	10274.00
5m10n3	**1740**	14352	1798	**12025**	1916	14471	1937	14783	1877	16225	1857	15707	1578.00	10909.00
5m10n4	**1628**	13360	1877	**11803**	1858	13814	1796	13658	1768	14835	1667	13548	1496.50	10493.00
5m10n5	1605	12920	1904	**12082**	1832	13282	1727	13495	1605	12960	**1595**	12484	1372.00	9101.00
10m10n1	**2011**	18487	2337	**17150**	2393	20219	2296	20147	2166	20271	2291	20262	1920.00	16425.00
10m10n2	**2147**	19352	2402	**18189**	2379	20572	2542	19944	2197	20035	2432	20040	1864.62	15213.00
10m10n3	**2042**	18281	2372	18441	2534	18467	2201	**18182**	**2042**	19504	2155	19494	1809.50	15266.00
10m10n4	**1911**	16852	2217	**16239**	2180	18284	2069	17399	2020	18092	2350	18870	1714.00	15166.50
10m10n5	2095	18099	**2005**	**16566**	2227	18269	2135	18237	2095	19256	2152	18805	1871.00	15219.00
10m15n1	**2338**	31937	2537	**30120**	2766	34472	2679	33615	2549	35767	2509	34063	2199.00	28108.00
10m15n2	**2566**	34467	2600	33996	2913	36137	2780	**33794**	2575	35807	2657	34993	2413.00	28712.50
10m15n3	**2515**	32593	2610	**30318**	2865	33441	2958	33425	2601	34711	2613	33899	2269.00	26776.50
10m15n4	**2474**	31312	2691	**30082**	2972	35350	2984	34169	2562	33470	2536	32766	2562.00	26108.00
10m15n5	**2469**	33559	2841	**30419**	2872	34241	2535	33715	2512	34410	2792	35480	2183.00	27804.00

bold: best solution

*: optimal solution

Table A.15: Test results for resumable operations and flexible unavailability periods placed at the end of their time windows with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**	*1254*	4043	939	4145	827	3681	875	*4210*	875	*4210*	825*	3356*
5m5n2	**1076**	4757	*1316*	*5571*	**1076**	**4687**	**1076**	4747	**1076**	4757	**1076**	4765	977*	4174*
5m5n3	1114	4579	*1410*	**4370**	1202	*5193*	1352	5158	1132	5055	**1100**	4479	1020*	3892*
5m5n4	**1217**	5045	1369	**4335**	*1419*	*5944*	1262	5320	**1217**	5157	1398	5910	1108*	4136*
5m5n5	**1211**	5045	**1211**	5000	1252	5203	1252	**4928**	1312	*5835*	*1392*	5559	1108*	4417*
5m10n1	1580	13512	1727	**10992**	1742	*13752*	*1750*	12184	**1540**	12748	1543	12789	1316	9484
5m10n2	**1525**	**12624**	1795	13242	1641	13320	*1925*	*13445*	1551	13236	**1525**	13091	1400	10364
5m10n3	**1740**	14352	1798	**12025**	1916	14471	*1937*	14783	1877	*16225*	1857	15707	1563	10710
5m10n4	**1628**	13360	*1877*	**11803**	1758	13701	1728	14084	1768	*14835*	1667	13548	1492	10439
5m10n5	1605	12920	*1904*	**12082**	1667	12824	1727	*13495*	1605	12960	**1595**	12484	1372	9101
10m10n1	**2011**	18487	2337	**17150**	*2393*	20219	2296	20147	2166	*20271*	2291	20262	1912	16397
10m10n2	**2147**	19352	2402	**18189**	2379	20572	*2542*	19944	2197	20035	2432	*20040*	1915	15190
10m10n3	**2042**	18281	*2372*	18441	2368	18642	2201	**18182**	**2042**	*19504*	2155	19494	1726	15261
10m10n4	**1911**	16852	2217	**16239**	2180	18284	2069	17399	2020	18092	*2350*	*18870*	1701	14876
10m10n5	2095	18099	**2005**	**16566**	*2167*	18329	2135	18237	2095	*19256*	2152	18805	1839	15273
10m15n1	**2338**	31937	2537	**30120**	*2739*	33748	2659	33375	2549	*35767*	2509	34063	2087	26762
10m15n2	**2566**	34467	2600	33996	*2802*	34798	2780	**33794**	2575	*35807*	2657	34993	2450	28685
10m15n3	**2515**	32593	2610	**30318**	2711	33091	*2958*	33702	2601	*34711*	2613	33899	4202	27221
10m15n4	**2474**	31312	*2691*	**30082**	2642	*33685*	2604	32459	2562	33470	2536	32766	4941	27264
10m15n5	**2469**	33559	2841	**30419**	*2872*	34241	2598	34277	2512	34410	2792	*35480*	2292	28000

*: optimal solution

∼: gap between the best solution and the MIP solution ≤ 10%

+: better solution than MIP one

Table A.16: Test results for resumable operations and flexible unavailability periods placed at the beginning of their time windows with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**910**	4004	*1160*	**3650**	1030	3970	1032	3974	**910**	*4076*	**910**	*4076*	825*	3356*
5m5n2	1080	4436	1046	*4926*	*1158*	4692	1062	4400	**977***	**4320**	996	4444	977*	4174*
5m5n3	1126	4590	1236	4562	*1270*	**4296**	**1042**	4438	1126	*5290*	1100	4488	1020*	3892*
5m5n4	1228	5089	1268	**4136**	1237	5031	*1287*	*5217*	**1157**	4917	1268	5102	1108*	4136*
5m5n5	**1160**	**4790**	1180	4904	*1291*	5180	*1291*	5180	1206	*5268*	1284	4946	1108*	4417*
5m10n1	1540	12472	1633	**10521**	*1843*	12587	1726	12152	**1533**	*12788*	1540	12526	1316	9484
5m10n2	**1460**	12308	*1650*	12595	1634	12496	1634	**11951**	1539	13120	1524	*13221*	1400	10364
5m10n3	**1740**	14836	*1917*	**12593**	1804	14869	1870	14637	1814	*15631*	1814	15364	1563	10710
5m10n4	**1617**	13261	*1772*	**11090**	1733	13522	1718	13474	1771	*14581*	1618	13244	1492	10439
5m10n5	**1486**	12189	1731	**11048**	1693	*13573*	*1797*	13507	1506	12645	**1486**	12047	1372	9101
10m10n1	**2011**	17983	*2307*	**17606**	2270	18499	2194	18940	2102	*19625*	2239	19597	1912	16397
10m10n2	**2078**	18770	2417	**18168**	2430	21406	2480	*21856*	2207	20165	*2633*	20665	1915	15190
10m10n3	**1902**	**17176**	2046	17485	*2284*	18428	2095	17940	2008	18782	2093	*18840*	1726	15261
10m10n4	**1927**	17305	2205	**16193**	2165	18593	*2327*	17605	2084	18655	2313	*19108*	1701	14876
10m10n5	**1977**	16923	2047	**16707**	2315	*18650*	*2510*	18516	2043	18348	2046	17724	1839	15273
10m15n1	**2377**	32069	2456	**28343**	*2874*	33148	2770	33936	2446	*34556*	2528	33424	2087	26762
10m15n2	**2460**	33557	2728	**33116**	*2930*	35754	2920	*36780*	2557	34671	2657	34764	2450	28685
10m15n3	**2341**	30993	2753	**30194**	2616	32558	*2808*	34511	2636	*35206*	2660	33684	4202	27221
10m15n4	**2441**	31767	2628	**30695**	*2912*	31550	2760	*33375*	2543	32738	2507	32807	4941	27264
10m15n5	**2396**	32893	2794	**30360**	2730	34046	*2868*	33394	2442	33615	2802	*35490*	2292	28000

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.17: Test results for resumable operations and flexible unavailability periods for order CAB in OIp procedure with OpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**845**	**3768**	1274	4143	959	4245	847	3781	895	4310	895	4310	825*	3356*
5m5n2	**1096**	4857	1336	5671	**1096**	**4777**	**1096**	4847	**1096**	4857	**1096**	4857	977*	4174*
5m5n3	1134	**4667**	1430	4735	**1062**	4685	**1062**	4685	1222	5293	1070	4671	1020*	3892*
5m5n4	**1237**	5261	1389	**4435**	1379	5691	1277	5405	**1237**	5261	1398	5910	1108*	4136*
5m5n5	1332	5659	1439	5547	**1280**	5329	1339	**5229**	1332	5659	1332	5499	1108*	4417*
5m10n1	1700	14064	1747	**11312**	1670	12637	1580	11531	**1560**	12952	1563	12794	1316	9484
5m10n2	**1545**	**13058**	1815	13513	1562	13249	1945	13645	1635	13665	**1545**	13221	1400	10364
5m10n3	**1754**	14564	1884	**12639**	2057	14165	1917	15295	1897	16425	1877	16019	1563	10710
5m10n4	**1648**	13520	1897	**11937**	1788	13974	1797	14510	1788	14864	1767	14864	1492	10439
5m10n5	1625	13200	1924	**12476**	1697	12921	1865	13565	1625	13102	**1615**	12644	1372	9101
10m10n1	**2031**	18753	2357	**17333**	2413	20419	2300	20271	2200	20365	2311	20462	1912	16397
10m10n2	**2167**	19710	2422	**18578**	2269	19198	2601	18865	2217	20235	2421	20862	1915	15190
10m10n3	2062	**18321**	2382	18587	2326	18576	2221	18382	**2005**	18974	2175	19825	1726	15261
10m10n4	**1990**	17147	2237	**16377**	2200	18604	2089	17542	2040	18356	2530	20326	1701	14876
10m10n5	2115	18299	**2025**	**16988**	2222	18324	2155	18437	2115	19456	2312	19125	1839	15273
10m15n1	2437	32479	**2257**	**28439**	2779	34682	2629	32390	2549	35127	2563	33897	2087	26762
10m15n2	2615	35232	2830	**33160**	2910	36453	2800	34581	**2585**	35535	2677	35183	2450	28685
10m15n3	**2535**	33018	2728	**30436**	2838	34561	2978	33933	2709	35365	2823	35143	4202	27221
10m15n4	**2494**	31605	2814	**30668**	2684	33714	3004	33923	2522	33119	2546	32581	4941	27264
10m15n5	**2489**	33973	2784	**30741**	2892	34947	2705	34690	2532	34710	2662	34798	2292	28000

bold: best solution

*: optimal solution

A.2 Machine based heuristics

Table A.18: Test results for non-preemptive operations and fixed unavailability periods with MOpH1 heuristic and different numbers of iterations.

Problem	100 iterations		1000 iterations		10000 iterations		100000 iterations		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	**895***	*4071*	**895***	3891	**895***	**3874**∼	**895***	**3874**∼	895*	3652*
5m5n2	**1112**∼	**4706**∼	**1112**∼	**4706**∼	**1112**∼	**4706**∼	**1112**∼	**4706**∼	1096*	4669*
5m5n3	**1070***	*4471*	**1070***	4411	**1070***	**4130**∼	**1070***	**4130**∼	1070*	4008*
5m5n4	*1235*	*4822*	**1205**∼	**4630**∼	**1205**∼	**4630**∼	**1205**∼	**4630**∼	1147*	4584*
5m5n5	**1202***	*4753*	**1202***	4683	**1202***	**4663***	**1202***	**4663***	1202*	4663*
5m10n1	**1420**∼	*11333*	**1420**∼	10831	**1420**∼	10877	**1420**∼	**10574**∼	1361*	9850*
5m10n2	*1545*	*12581*	1525	12252	**1485**∼	12023	**1485**∼	**11817**	1455	10637*
5m10n3	1677	12862	*1682*	*12887*	**1632**∼	12476	**1632**∼	**12057**∼	1607	11072
5m10n4	*1717*	*12267*	1617	12249	1586	11931	**1577**∼	**11683**∼	1537*	10630*
5m10n5	*1500*	*12140*	1495	11391	1495	10919	**1435**∼	**10898**	1415	9706*
10m10n1	*2151*	*18693*	2101	18503	2070	18275	**2047**∼	**17896**∼	1948	16878
10m10n2	*2182*	*18416*	2119	17724	2046	17418	**2022**∼	**17247**∼	1917*	15777*
10m10n3	1982	*17265*	*1992*	17260	**1912**∼	16689	**1912**∼	**16235**∼	1875	15352*
10m10n4	*2047*	*17792*	2004	17319	1931	16974	**1904**∼	**16571**∼	1772*	15374*
10m10n5	*2025*	*17948*	1985	17027	**1975**∼	16866	**1975**∼	**16559**∼	1975	15625*
10m15n1	*2504*	*31743*	2443	31199	2319	30590	**2274**∼	**29703**∼	2089	27199
10m15n2	*2611*	*33898*	2572	32458	2528	32673	**2480**∼	**32021**∼	2316	29186
10m15n3	*2515*	*32668*	2467	32030	2457	31274	**2401**∼	**30698**+	2261	u
10m15n4	*2534*	*32474*	2454	31256	2396	30864	**2312**∼	**30168**∼	2254	27423
10m15n5	*2499*	*33085*	2485	32479	2455	31672	**2422**∼	**31194**∼	2282	28609

bold: best solution

italic: worst solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.19: Test results for resumable operations and flexible unavailability periods with MOpH1 heuristic and different numbers of iterations.

Problem	100 iterations		1000 iterations		10000 iterations		100000 iterations		disjunctive MIP	
	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max} Min	$\sum C_i$ Min	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**∼	**825***	**3668**∼	**825***	**3668**∼	**825***	**3668**∼	825*	3356*
5m5n2	**1076**∼	**4537**∼	**1076**∼	**4537**∼	**1076**∼	**4537**∼	**1076**∼	**4537**∼	977*	4174*
5m5n3	**1034**	*4205*	**1034**∼	4177	**1034**∼	**4012**∼	**1034**∼	**4012**∼	1020*	3892*
5m5n4	*1215*	*4581*	1127	**4332**∼	**1108***	**4332**∼	**1108***	**4332**∼	1108*	4136*
5m5n5	**1182**∼	*4692*	**1182**∼	**4626**∼	**1182**∼	**4626**∼	**1182**∼	**4626**∼	1108*	4417*
5m10n1	*1442*	*11125*	1397	10917	1367	10513	**1360**∼	**10389**∼	1316	9484
5m10n2	*1501*	*12082*	1469	11792	1411	11537	**1409**∼	**11398**∼	1400	10364
5m10n3	*1652*	*12539*	1622	12385	1608	12318	**1578**∼	**11811**∼	1563	10710
5m10n4	*1678*	*12338*	1578	12035	1538	11793	**1518**∼	**11519**∼	1492	10439
5m10n5	*1535*	*11696*	1475	11095	1406	10678	**1405**∼	**10081**	1372	9101
10m10n1	*2073*	*18630*	2057	17678	1997	17844	**1911**+	**17269**∼	1912	16397
10m10n2	*2102*	*17720*	2097	17230	**1992**∼	16823	1995	**16766**∼	1915	15190
10m10n3	*1985*	*16759*	1901	16314	1869	16204	**1832**∼	**15962**∼	1726	15261
10m10n4	*2016*	*17138*	1920	16163	1873	16557	**1853**∼	**15975**∼	1701	14876
10m10n5	*1996*	*17057*	1955	16454	1923	16117	**1913**∼	**16041**∼	1839	15273
10m15n1	*2409*	*31555*	2360	30469	2289	29645	**2259**∼	**28903**∼	2087	26762
10m15n2	2520	*32686*	*2540*	32510	2494	31954	**2458**∼	**31641**∼	2450	28685
10m15n3	*2468*	*31865*	2403	30848	2412	30705	**2342**+	**29945**∼	4202	27221
10m15n4	*2454*	*30950*	2327	30857	2307	29477	**2302**+	**29439**∼	4941	27264
10m15n5	*2507*	*32529*	2476	31651	2392	31122	**2364**∼	**30541**∼	2292	28000

bold: best solution

italic: worst solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.20: CPU time for MOpH1 heuristic and different numbers of iterations.

Problem	Non-preemptive operations and fixed unavailability periods						Resumable operations and flexible unavailability periods					
	MOpH1				MIP		MOpH1				MIP	
	100 iterations	1000 iterations	10000 iterations	100000 iterations	C_{max}	$\sum C_i$	100 iterations	1000 iterations	10000 iterations	100000 iterations	C_{max}	$\sum C_i$
5m5n1	0	0.12	0.51	4.96	0.09	0.04	0	0.06	0.51	5.22	3.55	0.90
5m5n2	0.01	0.04	0.48	5.03	0.02	0.01	0.01	0.07	0.53	5.27	1.23	0.12
5m5n3	0	0.04	0.48	5.00	0.23	0.04	0	0.04	0.51	5.32	1.07	1.98
5m5n4	0	0.04	0.48	4.94	0.27	0.06	0	0.04	0.53	5.27	0.89	1.04
5m5n5	0.01	0.04	0.48	5.00	0.51	0.07	0	0.04	0.53	5.28	10.82	2.37
5m10n1	0.01	0.15	1.43	14.61	43.09	672.04	0.01	0.15	1.51	15.08	3496.37	3600
5m10n2	0.01	0.14	1.46	14.78	1519.4	167.89	0.01	0.15	1.52	15.22	3449.11	3600
5m10n3	0.01	0.15	1.43	14.75	1939.35	3600	0.01	0.15	1.52	15.17	3489.86	3600
5m10n4	0.01	0.14	1.41	14.57	1324.87	565	0.01	0.14	1.48	14.96	3498.93	3600
5m10n5	0.01	0.14	1.43	14.60	174.72	1566.81	0.01	0.15	1.49	14.89	3451.59	3600
10m10n1	0.06	0.67	6.80	69.70	2073.96	7.40	0.06	0.68	6.78	67.65	3561.88	3600
10m10n2	0.06	0.67	6.83	69.67	664.17	547.82	0.06	0.67	6.77	67.62	3535.8	3600
10m10n3	0.07	0.68	6.81	69.63	466.46	646.60	0.07	0.67	6.78	67.75	3552.65	3600
10m10n4	0.06	0.68	6.84	69.91	669.87	3600	0.06	0.68	6.81	68.11	3552.26	3600
10m10n5	0.07	0.68	6.84	69.90	1814.3	1237.39	0.07	0.68	6.81	67.86	3535.78	3600
10m15n1	0.17	1.71	17.05	174.20	2708.88	3600	0.15	1.65	16.58	165.73	3573.83	3600
10m15n2	0.17	1.70	17.03	173.87	1893.73	3600	0.15	1.66	16.50	165.31	3532.5	3600
10m15n3	0.17	1.71	17.39	174.51	2645.88	3600	0.17	1.65	16.58	165.97	3555.99	3600
10m15n4	0.18	1.71	17.08	173.97	2679.33	3600	0.15	1.63	16.63	165.62	3542.84	3600
10m15n5	0.17	1.70	16.98	174.05	2807.98	3600	0.17	1.65	16.55	165.57	3568.95	3600

Table A.21: Test results for non-preemptive operations and fixed unavailability periods with MOpH2 heuristic and different rules.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	914	4094	1035	*4327*	*1114*	4185	1028	**3947**∼	**895***	4310	**895***	4310	895*	3652*
5m5n2	**1112**∼	**4706**∼	1226	*5861*	**1112**∼	**4706**∼	*1213*	4823	**1112**∼	**4706**∼	**1112**∼	**4706**∼	1096*	4669*
5m5n3	1134	4929	*1349*	4957	**1122**∼	**4779**	**1122**∼	**4779**	1332	*5564*	1332	*5564*	1070*	4008*
5m5n4	**1255**	5355	*1389*	**4930**	1375	5487	**1255**	5355	1375	*5725*	1375	*5725*	1147*	4584*
5m5n5	*1332*	*5566*	*1332*	5529	**1280**∼	**5064**∼	**1280**∼	**5064**∼	*1332*	*5566*	*1332*	*5566*	1202*	4663*
5m10n1	1700	*14414*	*1987*	**11876**	1711	14274	1700	13024	**1560**	12992	**1560**	12992	1361*	9850*
5m10n2	**1635**	**12929**	*1925*	*14340*	**1635**	13621	**1635**	13325	**1635**	14286	**1635**	14286	1455	10637*
5m10n3	1897	16077	**1884**	**12782**	1887	14874	1887	14124	*1897*	*16117*	*1897*	*16117*	1607	11072
5m10n4	**1638**∼	13762	*1947*	**11839**	1757	13873	1738	13458	1738	*14891*	1738	*14891*	1537*	10630*
5m10n5	**1625**	13530	**1625**	**11457**	*1675*	*13701*	*1675*	13084	**1625**	13540	**1625**	13540	1415	9706*
10m10n1	**2141**∼	**19697**	2377	19835	*2587*	*22015*	2406	20159	2251	20749	2251	20749	1948	16878
10m10n2	2512	*21940*	**2296**	**17087**∼	2362	18579	*2542*	20893	2392	20921	2392	20921	1917*	15777*
10m10n3	**2072**	18884	*2472*	**18165**	2275	*19892*	2455	19258	2100	19352	2100	19352	1875	15352*
10m10n4	**2110**	18640	*2529*	18364	2224	18953	2154	**17638**	2160	*19091*	2160	*19091*	1772*	15374*
10m10n5	2155	18870	**2055**∼	**16731**∼	*2527*	*19847*	2405	18704	2115	18465	2115	18465	1975	15625*
10m15n1	**2420**	32415	2859	**29325**∼	*3208*	*34933*	2557	34139	2566	34090	2566	34090	2089	27199
10m15n2	2700	35860	2947	**34293**	*3030*	*37319*	2815	34590	**2533**∼	34577	**2533**∼	34577	2316	29186
10m15n3	2583	33839	*2827*	**31147**+	2647	*34457*	**2551**	32589	2663	34379	2663	34379	2261	u
10m15n4	**2494**	33055	*2804*	**32854**	2762	33243	2674	33096	2662	*33940*	2662	*33940*	2254	27423
10m15n5	**2515**∼	33813	2635	**31267**∼	2702	34560	*2722*	33186	2522	*35256*	2522	*35256*	2282	28609

bold: best solution

italic: worst solution

*: optimal solution

u: unknown solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.22: Test results for resumable operations and fixed unavailability periods with MOpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	895	3959	**845***	**3738**	1085	4124	1023	3889	895	4117	895	4117	845*	3577*
5m5n2	**1080**	**4482**	1137	5453	**1080**	**4482**	**1080**	**4482**	**1080**	**4482**	**1080**	**4482**	997*	4385*
5m5n3	1126	**4610**	1296	4816	**1110**	4640	**1110**	4640	1126	5008	1126	5008	1020*	3912*
5m5n4	1387	5865	**1268**	**4328**	1338	5559	1387	5865	1338	5559	1338	5559	1135*	4136*
5m5n5	**1168**	4992	1248	5180	1382	5269	1382	**4964**	**1168**	4992	**1168**	4992	1108*	4443*
5m10n1	**1560**	12572	1633	**10620**	1583	12924	1647	12671	**1560**	12654	**1560**	12654	1322	9610
5m10n2	**1460**	12322	1794	13388	1545	12698	1628	**11806**	1524	13281	1524	13281	1400	10402
5m10n3	**1770**	15126	1849	**12352**	1887	14972	1887	13862	1834	15636	1834	15636	1583	11130
5m10n4	1697	13760	1842	**11444**	1817	14563	1788	14044	**1621**	13420	**1621**	13420	1492	10526
5m10n5	**1486**	12209	1697	**10868**	1675	13122	1681	12776	1498	12853	1498	12853	1386	9407
10m10n1	**2151**	**18888**	2382	19229	2504	21620	2644	20063	2152	19910	2152	19910	1907	16335
10m10n2	2284	19990	**2132**	**16692**	2528	18909	2643	19485	2252	19659	2252	19659	1822	15353
10m10n3	**2000**	17678	2160	**17548**	2320	18818	2246	18456	2013	18624	2013	18624	1815	15349
10m10n4	**1978**	17142	2181	**16799**	2154	17934	1998	17610	2069	18152	2069	18152	1755	15423
10m10n5	**1977**	17570	2047	**17249**	2359	19615	2190	18292	2093	18500	2093	18500	1870	15669
10m15n1	**2377**	32624	2473	**28527**	2943	33701	2669	32923	2500	33276	2500	33276	2310	27010
10m15n2	2566	36211	2922	34070	2972	35978	2725	**33971**	**2550**	35314	**2550**	35314	2460	29251
10m15n3	**2493**	31739	2615	**30850**	2838	33171	2750	32904	2576	33447	2576	33447	2313	27196
10m15n4	**2441**	31411	2595	**30610**	2492	33564	2717	34027	2520	32230	2520	32230	6958	27530
10m15n5	**2406**	33351	2669	**31546**	2690	36073	2919	32587	2595	34417	2595	34417	6392	27601

bold: best solution

*: optimal solution

Table A.23: Test results for non-resumable operations and fixed unavailability periods with MOpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	914	4094	1035	4327	1164	4149	1028	**3947**	**895***	4310	**895***	4310	895*	3652*
5m5n2	**1112**	**4706**	1226	5861	**1112**	**4706**	**1112**	4766	**1112**	**4706**	**1112**	**4706**	1096*	4669*
5m5n3	1134	4929	1349	4957	**1122**	**4689**	**1122**	**4689**	1332	5564	1332	5564	1070*	4008*
5m5n4	**1375**	5839	1389	**4930**	1389	5693	1529	6137	**1375**	5839	**1375**	5839	1147*	4584*
5m5n5	1332	5654	1332	5529	**1280**	**5064**	1400	5565	1332	5704	1332	5704	1202*	4663*
5m10n1	1700	14491	1987	**11876**	1650	13403	1630	12902	**1560**	13323	**1560**	13323	1361	9850
5m10n2	1635	13596	1944	15227	1785	**13114**	**1605**	12803	1635	14286	1635	14286	1455	10643
5m10n3	1897	16077	**1884**	**12782**	1887	15021	1887	14180	1897	16117	1897	16117	1637	11072
5m10n4	1788	14489	1947	**11839**	1817	14563	1788	14559	**1738**	14891	**1738**	14891	1537	10630
5m10n5	1685	13908	**1625**	**11457**	1675	13514	1727	13490	1685	14108	1685	14108	1415	9706
10m10n1	**2228**	20130	2377	**19835**	2557	22173	2531	19873	2273	20527	2273	20527	1987	17117
10m10n2	2392	21350	**2296**	**17087**	2362	19613	2622	20233	2442	21131	2442	21131	1917	15777
10m10n3	**2211**	19452	2472	**18165**	2275	19899	2281	19580	2230	19826	2230	19826	1875	15644
10m10n4	**2095**	18770	2427	18767	2338	18621	2157	**17931**	2329	19328	2329	19328	1772	15374
10m10n5	2155	18870	**2055**	**16731**	2695	20359	2405	19414	2115	18465	2115	18465	1933	15719
10m15n1	**2500**	32856	2859	**29325**	3054	36173	2889	32908	2769	35891	2769	35891	2209	26886
10m15n2	**2702**	37083	2760	**33866**	3078	38266	2900	35572	2740	37280	2740	37280	2326	u
10m15n3	2621	34691	2827	**31147**	2873	33201	**2581**	32547	2663	34379	2663	34379	2213	28308
10m15n4	**2542**	32831	2726	**31738**	2754	35268	2927	35071	2662	34344	2662	34344	2212	27763
10m15n5	**2515**	33813	2635	**31267**	2958	37688	2762	34907	2582	36555	2582	36555	2342	19226.97

bold: best solution

*: optimal solution

u: unknown solution

Table A.24: Test results for semi-resumable operations and fixed unavailability periods with MOpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**895**	4011.75	981.25	4024.75	1152	4111.5	1027.25	**3919.75**	**895**	4242.87	**895**	**4242.87**	866.62*	3613.00*
5m5n2	**1096**	**4604.5**	1170	5580	**1096**	**4604.5**	**1096**	4634.25	**1096**	**4604.5**	**1096**	**4604.5**	1042.25*	4601.00*
5m5n3	1133	**4640**	1437.25	4974.25	**1116**	4664.5	**1116**	4664.5	1277	5387	1277	5387	1046.50*	3938.00*
5m5n4	1346.5	5721	**1338**	**4643.5**	1346.5	5721	1346.5	5721	1346.5	5721	1346.5	5721	1137.00*	4307.00*
5m5n5	1272	5360.5	1309.5	5674	**1239.75**	**4900.25**	1354	5321	1272	5360.5	1272	5360.5	1159.50*	4597.50*
5m10n1	**1560**	13124	1700.75	**11012.5**	1590	13130	1650	12895	**1560**	12798.43	**1560**	12798.43	1378.50	9843.00
5m10n2	**1535.25**	12975.25	1832.75	13711.75	1768	**12875.5**	1545	12325.5	1606.5	13952.75	1606.5	13952.75	1452.75	10440.50
5m10n3	**1849.5**	15677.5	1856.5	**12624**	1887	15009	1887	14041.5	1865.5	15894.25	1865.5	15894.25	1625.50	11106.00
5m10n4	1756.25	14226.25	1913.5	**12059.5**	1817	14563	1788	14228.5	**1638**	13658	**1638**	13658	1503.50	10588.00
5m10n5	**1548.25**	12804.5	1686	**11237.5**	1675	13173.625	1725.5	13307.5	**1548.25**	13285.75	**1548.25**	13285.75	1406.00	9515.00
10m10n1	**2117.75**	19636.87	2271	**18279**	2410.5	21376.5	2562.25	20623.37	2177	19961.5	2177	19961.5	1906.00	16806.00
10m10n2	2440.75	21156.5	**2262**	**16784.5**	2351.5	19576.5	2492	19643	2315.25	20353	2315.25	20353	1957.50	15572.70
10m10n3	**1992**	**17510**	2321	18034	2255	19225.25	2375.5	19042.25	2062	19318	2062	19318	1825.25	15352.00
10m10n4	**2009**	17862.5	2295	**17418.25**	2120	18526	2179.5	17710.5	2086	18309.5	2086	18309.5	1780.00	15411.20
10m10n5	2132	19145.75	2080.5	**17508.5**	2749	19673	2410.12	18532.75	**2115**	18734.25	**2115**	18734.25	1938.37	15595.00
10m15n1	**2405.25**	32908	2367.25	**29794.37**	3210	37449	2613	32358	2496.5	34479.75	2496.5	34479.75	2107.00	26993.20
10m15n2	2703.12	36681.37	2927.5	**34614.87**	2992	37572	3015.5	37003.5	**2572.25**	36062.25	**2572.25**	36062.25	2366.00	u
10m15n3	**2548.62**	33302.93	2752	**31139.31**	2838.5	33217	2585.5	32496.5	2640.87	34128.81	2640.87	34128.81	2271.00	u
10m15n4	**2536.37**	32299.5	2781.5	**31492.25**	2735	35055.5	2847	34308.5	2606.5	33496.5	2606.5	33496.5	2214.00	27277.20
10m15n5	**2489**	34384.5	2819	33619.125	2938	33851.25	2829.5	**32532.25**	2654.5	35490	2654.5	35490	2272.25	28805.20

bold: best solution

*: optimal solution

u: unknown solution

Table A.25: Test results for non-preemptive operations and flexible unavailability periods with MOpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**	**825***	**3668**	1005	4109	957	3759	875	4210	875	4210	825*	3552*
5m5n2	**1076***	**4605**	1123	5279	**1076***	**4605**	**1076***	**4605**	**1076***	**4605**	**1076***	**4605**	1076*	4412*
5m5n3	1114	**4579**	1462	5010	1110	4617	**1070**	4639	1242	5120	1242	5120	1076*	3968*
5m5n4	**1235**	5255	1369	**4335**	1355	5625	**1235**	5255	1355	5625	1355	5625	1108*	4242*
5m5n5	1312	5466	1312	5429	1211	**4975**	**1211**	**4975**	1312	5466	1312	5466	1182*	4565*
5m10n1	1680	13937	1727	**11155**	1551	12390	1596	11393	**1540**	12748	**1540**	12748	1300*	9535*
5m10n2	**1475**	12550	1825	13685	1905	12644	1740	**11917**	1615	13741	1615	13741	1400	10256*
5m10n3	1807	14871	1864	**12481**	1867	14730	**1794**	14093	1877	15917	1877	15917	1578	10997
5m10n4	**1618**	13562	1877	**11994**	1708	13501	1748	13923	**1618**	13562	**1618**	13562	1492	10491*
5m10n5	1605	12920	1725	12078	1655	12253	**1577**	**11886**	1605	13039	1605	13039	1372*	9101
10m10n1	2237	19680	2417	**18316**	2491	20760	2640	20306	**2156**	19958	**2156**	19958	1859	16085
10m10n2	2434	21109	**2202**	**16433**	2311	18283	2357	19690	2245	19570	2245	19570	1839*	15200*
10m10n3	**1972**	17850	2212	**17557**	2255	19291	2362	17630	2125	19351	2125	19351	1773*	15164
10m10n4	**2070**	17968	2270	**16404**	2185	17287	2115	17874	2126	18785	2126	18785	1690*	15155
10m10n5	2185	18479	2005	**16683**	2291	18503	**1995**	17204	2095	18105	2095	18105	1833	15136
10m15n1	**2400**	32129	2718	**29530**	2798	34215	2829	33314	2451	32918	2451	32918	2103	27221
10m15n2	**2493**	34513	2853	**33350**	3202	37545	2669	36207	2513	34277	2513	34277	2388	27977
10m15n3	2581	32677	2783	**30338**	2836	34515	2652	32097	**2539**	32705	**2539**	32705	2221	u
10m15n4	**2474**	31304	2656	**30656**	2738	33779	2574	32060	2571	33176	2571	33176	2120	27143
10m15n5	**2469**	34094	2765	**31879**	2824	35469	2809	33817	2489	34669	2489	34669	2225	28312

bold: best solution

*: optimal solution

u: unknown solution

Table A.26: Test results for resumable operations and flexible unavailability periods with MOpH2 heuristic and different rules.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**∼	**825***	**3668**∼	1005	4109	957	3759	875	*4210*	875	*4210*	825*	3356*
5m5n2	**1076**∼	**4605**∼	*1123*	*5279*	**1076**∼	**4605**∼	**1076**∼	**4605**∼	**1076**∼	**4605**∼	**1076**∼	**4605**∼	977*	4174*
5m5n3	1114	**4579**	*1462*	5010	1110	4617	**1070**∼	4639	1242	*5120*	1242	*5120*	1020*	3892*
5m5n4	**1235**	5255	*1369*	**4335**∼	1355	*5625*	**1235**	5255	1355	*5625*	1355	*5625*	1108*	4136*
5m5n5	1312	*5466*	1312	5429	**1211**∼	**4975**	**1211**	**4975**	1312	*5466*	*1312*	*5466*	1108*	4417*
5m10n1	1680	*13937*	*1727*	**11155**	1551	12390	1596	11393	**1540**	12748	**1540**	12748	1316	9484
5m10n2	**1475**∼	12550	1825	13685	*1905*	12644	1740	**11917**	1615	*13741*	1615	*13741*	1400	10364
5m10n3	1807	14871	1864	**12481**	1867	14730	**1794**	14093	*1877*	*15917*	*1877*	*15917*	1563	10710
5m10n4	**1618**∼	13562	*1877*	**11994**	1708	13501	1748	*13923*	**1618**	13562	**1618**	13562	1492	10439
5m10n5	1605	12920	*1725*	12078	1655	12253	**1577**	**11886**	1605	*13039*	1605	*13039*	1372	9101
10m10n1	2237	19680	2417	**18316**	2491	*20760*	*2640*	20306	**2156**	19958	**2156**	19958	1912	16397
10m10n2	*2434*	*21109*	**2202**	**16433**∼	2311	18283	2357	19690	2245	19570	2245	19570	1915	15190
10m10n3	**1972**	17850	2212	**17557**	2255	19291	*2362*	17630	2125	*19351*	2125	*19351*	1726	15261
10m10n4	**2070**	17968	*2270*	**16404**∼	2185	17287	2115	17874	2126	*18785*	2126	*18785*	1701	14876
10m10n5	2185	18479	2005	**16683**∼	*2291*	*18503*	**1995**∼	17204	2095	18105	2095	18105	1839	15273
10m15n1	**2400**	32129	2718	**29530**∼	2798	*34215*	*2829*	33314	2451	32918	2451	32918	2087	26762
10m15n2	**2493**∼	34513	2853	**33350**	*3202*	*37545*	2669	36207	2513	34277	2513	34277	2450	28685
10m15n3	2581	32677	2783	**30338**	*2836*	*34515*	2652	32097	**2539**+	32705	**2539**	32705	4202	27221
10m15n4	**2474**+	31304	2656	**30656**	*2738*	*33779*	2574	32060	2571	33176	2571	33176	4941	27264
10m15n5	**2469**∼	34094	2765	**31879**	*2824*	*35469*	2809	33817	2489	34669	2489	34669	2292	28000

bold: best solution

italic: worst solution

*: optimal solution

∼: gap between the best solution and the MIP solution $\leq 10\%$

+: better solution than MIP one

Table A.27: Test results for non-resumable operations and flexible unavailability periods with MOpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**	**825***	**3668**	1005	4109	957	3759	875	4210	875	4210	825*	3352*
5m5n2	**1076***	**4605**	1123	5279	**1076***	**4605**	**1076***	**4605**	**1076***	**4605**	**1076***	**4605**	1076*	4412*
5m5n3	1114	**4579**	1462	5010	1110	4617	**1070**	4639	1242	5120	1242	5120	1034*	3968*
5m5n4	**1235**	5255	1369	**4335**	1355	5625	**1235**	5255	1355	5625	1355	5625	1108*	4242*
5m5n5	1312	5466	1312	5429	**1211**	**4975**	**1211**	**4975**	1312	5466	1312	5466	1182*	4565*
5m10n1	1680	13937	1727	**11155**	1551	12390	1596	11393	**1540**	12748	**1540**	12748	1302	9585
5m10n2	**1475**	12550	1825	13685	1905	12644	1740	**11917**	1615	13741	1615	13741	1400	10274
5m10n3	1807	14871	1864	**12481**	1867	14730	**1794**	14093	1877	15917	1877	15917	1578	10558
5m10n4	**1618**	13562	1877	**11994**	1708	13501	1748	13923	**1618**	13562	**1618**	13562	1492	10493
5m10n5	1605	12920	1725	12078	1655	12253	**1577**	**11886**	1605	13039	1605	13039	1377	9292
10m10n1	2237	19680	2417	**18316**	2491	20760	2640	20306	**2156**	19958	**2156**	19958	1912	16297
10m10n2	2434	21109	**2202**	**16433**	2311	18283	2357	19690	2245	19570	2245	19570	1874	15213
10m10n3	**1972**	17850	2212	**17557**	2255	19291	2362	17630	2125	19351	2125	19351	1773	15278
10m10n4	**2070**	17968	2270	**16404**	2185	17287	2115	17874	2126	18785	2126	18785	1690	15045
10m10n5	2185	18479	2005	**16683**	2291	18503	**1995**	17204	2095	18105	2095	18105	1880	15347
10m15n1	**2400**	32129	2718	**29530**	2798	34215	2829	33314	2451	32918	2451	32918	2137	u
10m15n2	**2493**	34513	2853	**33350**	3202	37545	2669	36207	2513	34277	2513	34277	2395	28004
10m15n3	2581	32677	2783	**30338**	2836	34515	2652	32097	**2539**	32705	**2539**	32705	2403	27669
10m15n4	**2474**	31304	2656	**30656**	2738	33779	2574	32060	2571	33176	2571	33176	2166	28015
10m15n5	**2469**	34094	2765	**31879**	2824	35469	2809	33817	2489	34669	2489	34669	2146	27763

bold: best solution

*: optimal solution

u: unknown solution

Table A.28: Test results for semi-resumable operations and flexible unavailability periods with MOpH2 heuristic.

Problem	Rule (1)		Rule (2)		Rule (3)		Rule (4)		Rule (5)		Rule (6)		disjunctive MIP	
	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$	C_{max}	$\sum C_i$
5m5n1	**825***	**3668**	**825***	**3668**	1005	4109	957	3759	875	4210	875	4210	825.00*	3366.50*
5m5n2	**1076**	**4605**	1123	5279	**1076**	**4605**	**1076**	**4605**	**1076**	**4605**	**1076**	**4605**	1000.00*	4218.25*
5m5n3	1114	**4579**	1462	5010	1110	4617	**1070**	4639	1242	5120	1242	5120	1031.50*	3918.00*
5m5n4	**1235**	5255	1369	**4335**	1355	5625	**1235**	5255	1355	5625	1355	5625	1108.00*	4220.50*
5m5n5	1312	5466	1312	5429	**1211**	**4975**	**1211**	**4975**	1312	5466	1312	5466	1145.00*	4528.00
5m10n1	1680	13937	1727	**11155**	1551	12390	1596	11393	**1540**	12748	**1540**	12748	1339.25	9535.00
5m10n2	**1475**	12550	1825	13685	1905	12644	1740	**11917**	1615	13741	1615	13741	1400.00	10274.00
5m10n3	1807	14871	1864	**12481**	1867	14730	**1794**	14093	1877	15917	1877	15917	1578.00	10909.00
5m10n4	**1618**	13562	1877	**11994**	1708	13501	1748	13923	**1618**	13562	**1618**	13562	1496.50	10493.00
5m10n5	1605	12920	1725	12078	1655	12253	**1577**	**11886**	1605	13039	1605	13039	1372.00	9101.00
10m10n1	2237	19680	2417	**18316**	2491	20760	2640	20306	**2156**	19958	**2156**	19958	1920.00	16425.00
10m10n2	2434	21109	**2202**	**16433**	2311	18283	2357	19690	2245	19570	2245	19570	1864.62	15213.00
10m10n3	**1972**	17850	2212	**17557**	2255	19291	2362	17630	2125	19351	2125	19351	1809.50	15266.00
10m10n4	**2070**	17968	2270	**16404**	2185	17287	2115	17874	2126	18785	2126	18785	1714.00	15166.50
10m10n5	2185	18479	2005	**16683**	2291	18503	**1995**	17204	2095	18105	2095	18105	1871.00	15219.00
10m15n1	**2400**	32129	2718	**29530**	2798	34215	2829	33314	2451	32918	2451	32918	2199.00	28108.00
10m15n2	**2493**	34513	2853	**33350**	3202	37545	2669	36207	2513	34277	2513	34277	2413.00	28712.50
10m15n3	2581	32677	2783	**30338**	2836	34515	2652	32097	**2539**	32705	**2539**	32705	2269.00	26776.50
10m15n4	**2474**	31304	2656	**30656**	2738	33779	2574	32060	2571	33176	2571	33176	2562.00	26108.00
10m15n5	**2469**	34094	2765	**31879**	2824	35469	2809	33817	2489	34669	2489	34669	2183.00	27804.00

bold: best solution

*: optimal solution

u: unknown solution

Appendix B

References

[AA06] H. Allaoui, A. Artiba, (2006). *Scheduling two-stage hybrid flow shop with availability constaints.* Computers and Operations Research, 33, 5, 1399-1419.

[AAER06] H. Allaoui, A. Artiba, S.E. Elmaghraby, F. Riane, (2006). *Scheduling of a two-machine flow shop with availability constaints on the first machine.* International Journal of Production Economics, 99, 1-2, 16-27.

[AAR03] H. Allaoui, H. Artiba, F. Riane, (2003). *Scheduling of two-machine flow-shops with availability constraints.* Dans : Proceedings of 18th ORBEL conference. Brussels, Belgium.

[ABFRK89] I. Adiri, J. Bruno, E. Frostig, A.H.G. Rinnooy Kan, (1989). *Single machine flow-time scheduling with a single breakdown.* Acta Informatica, 26, 679-696.

[AC91] D. Applegate, W. Cook, (1991). *A Computational Study of the Job-shop Scheduling Problem.* ORSA Journal of Computing, 3, 149-156.

[AF55] S.B. Akers, J. Friedman, (1955). *A Non-numerical Approach to Production Scheduling Problems.* Operations Research, 3, 429-442.

[Agg02] R. Aggoune, (2002). *Ordonnancement d'ateliers sous contraintes de disponibilités des machines.* Thèse de PhD. Université de Metz, France.

[Agg04b] R. Aggoune, (2004). *Une procédure par séparation et évaluation pour l'ordonnancement d'un job shop sous contraintes de disponibilité.* Dans: Proceeding of 5e Conférence Francophone de Modélisation et SIMulation (MOSIM'04), Nantes.

[AHS00] J. M. Van Den Akker, C. A. J. Hurkens, M. W. P. Savelbergh, (2000). *Time-indexed formulations for machine scheduling problems: Column generation.* INFORMS Journal on Computing, 12, 2, 111-124.

[All04] H. Allaoui, (2004). *Hybrid Flow-shop Scheduling with Maintenance Constraints: Complexity, Algorithms and Application.* Thèse de PhD, Université Pierre Marie Curie et Faculté Universitaires Catholiques de Mons.

[AP06] R. Aggoune, M.C. Portmann, (2006). *Flow shop scheduling problem with limited machine availability: A heuristic approach.* International Journal of Production Economics, 99, 4-15.

[Bak74] K.R. Baker, (1974). *Introduction to Sequencing and Scheduling.* Wiley, New York.

[BBFKS01] J. Blazewicz, J. Breit, P. Formanowicz, W. Kubiak, G. Schmidt, (2001). *Heuristic algorithms for the two-machine flowshop Problem with limited machine availability.* Omega Journal, 29, 599-608.

[BDFKS00] J. Blazewicz, M. Drozdowski, P. Formanowicz, W. Kubiak, G. Schmidt, (2000), *Scheduling preemptable tasks on parallel processors with limited availability.* Parallel Computing 26(9); 1195-1211.

[BDODM03] J. Blazewicz, P. Dell'Olmo, M. Drozdowski, P. Maczka, (2003), *Scheduling multiprocessor tasks on parallel processors with limited availability.* European Journal of Operational Research 149; 377-389.

[Bel57] R. Bellman, (1957). *Dynamic Programming.* Princeton University Press.

[Ben05] F. Benbouzid, (2005). *Une Contribution l'étude de la performance et de la robustesse des ordonnancements conjoints Production/Maintenance - Cas Flow-Shop.* Thèse de PhD, Université de Franche Comté, France.

[BEPSW96] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, J. Weglarz, (1996). *Scheduling Computer and Manufacturing Processes.* Springer, Berlin.

[BEPSW07] J. Blazewicz, K.H. Ecker, E. Pesch, G. Schmidt, J. Weglarz, (2007). *Handbook on scheduling from Theory to Applications* . International Handbooks on Information systems.

[BFKPS00] J. Blazewicz, P. Formanowicz, W. Kubiak, M. Przysucha, G. Schmidt, (2000). *Parallel Branch and Bound Algorithms for the Two-machine Flow Shop Problem with Limited Machine Availability.* Bulletin of the Polish Academy of Sciences, Technical Sciences, 48, 1, 105-115.

[BJNSV98] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, P.H. Vance, (1998) *Branch-and-Price: Column Generation for Solving Huge Integer Programs.* Operations Research, 46, 3, 316-329.

[BLSS02] O. Braun, T.C. Lai, G. Schmidt, Y.N. Sotskov (2002). *Stability of Johnsons schedule with respect to limited machine availability.* International Journal of Production Research, 40, 17, 4381-4400.

[BLSV98] E. Balas, G. Lancia, P. Serafini, A. Vazacopoulos, (1998). *Job shop scheduling with deadlines*. Journal of Combinatorial Optimization, vol. 1, n 4, pp. 329-353.

[Bre00] J. Breit, (2000). *Heuristische Ablaufplanungsverfahren fr Flowshops und Openshops mit beschrnkt verfgbaren Prozessoren*, Ph.D. Thesis, University of Saarland, Saarbrcken.

[Bre04] J. Breit, (2004). *An improved approximation algorithm for two-machine flow shop scheduling with an availability constraint*. Information Processing Letters 90, 273-278.

[Bre06] J. Breit, (2006). *A polynomial-time approximation scheme for the two-machine flow shop scheduling problem with an availability constraint*. Computers & operations, 33, 8, 2143-2153.

[Bru98] P. Brucker, (1998). *Scheduling Algorithms*. Springer-Verlag, Berlin Heidelberg.

[BSS01a] O. Braun, G. Schmidt, Y. Sotskov, (2001). *Stability of Jonson's Schedule with limited Machine availability*. Actes de la 3e conférence francophone de MOdélisation et de SIMulation (MOSIM'01), Troyes, Conception, analyse et gestion des systèmes industriels, SCS European Publishing House, 2, 683-687.

[BSS01b] J. Breit, G. Schmidt, V.A. Strusevich, (2001). *Two-machine open shop scheduling with an availability constraint*. Operations Research Letters, 29, 65-77.

[BSS03] J. Breit, G. Schmidt, V.A. Strusevich, (2003).*Non-Preemptive Two-Machine Open Shop Scheduling with Non-availability Constraints*. Mathematical methods of operations research, 57, 2, 217-234.

[CBB03] C. Canon, J.-C. Billaut, J.-L. Bouquard, (2003), *The one-machine sequencing problem with availability constraints*. Technical Report 271, Laboratoire d'Informatique de Université de Tours; Tours (France).

[CC88] J. Carlier, P. Chretienne, (1988). *Les Problèmes dordonnancement*. Masson, Paris, France.

[CDM91] A. Colorni, M. Dorigo, V. Maniezzo, (1991). *An Investigation of Some Properties of an Ant Algorithm*. In Parallel Problem Solving from Nature 2, R. Mnner and B. Manderick eds, North-Holland: Amsterdam, 509-520.

[Che06] W.J. Chen, (2006), *Minimizing total flow time in the single-machine scheduling problem with periodic maintenance*. The Journal of the Operational Research Society, 57, 4, 410-415.

[Che07] W.J. Chen, (2007), *Scheduling of jobs and maintenance in a textile company*. International Journal of Advanced Manufacturing Technology, 31, 737-742.

[CL03a] T.C.E. Cheng, Z. Liu, (2003). *3/2-approximation for two-machine no-wait flowshop scheduling with availability constraints*. Information Processing Letters, 88, 161-165.

[CL03b] T.C.E. Cheng, Z. Liu, (2003). *Approximability of two-machine no-wait flowshop scheduling with availability constraints*. Operations Research Letters, 31, 319-322.

[CMM67] R.W. Conway, W.L. Maxwell, L.W. Miller, (1967). *Theory of Scheduling*. Addison Wesley, Reading, Mass., USA.

[Coo71] S.A. Cook, (1971). *The Complexity of Theorem Proving Procedures*. Proceedings of the Third Annual ACM Symposium on the Theory of Computing, Association of Computing Machinery, New York, 151-158.

[CW99] T.C.E. Cheng, G. Wang, (1999). *Two-machine flowshop scheduling with consecutive availability constraints*. Information Processing Letters, 71, 2, 49-54.

[CW00] T.C.E. Cheng, G. Wang, (2000). *An improved heuristic for the two-machine flowshop scheduling with an availability constraint*. Operations Research Letters, 26, 223-229.

[CWC06] F.T.S. Chan, T.C. Wong, L.Y. Chan, (2006), *Flexible job-shop scheduling problem under resource Constraints*. International Journal of Production Research, 44, 11, 2071-2089.

[DAM03] S. Demassey, C. Artigues, Ph. Michelon (2003). *AConstraint-propagation-based cutting planes: An application to the resource-constrained project scheduling problem*. INFORMS Journal on Computing, 17 (1), 52-65.

[Dan51] G.B. Dantzig, (1951). *Application of the simplex method to a transportation problem, in Activity analysis of production and allocation*. T.C. Koopmans eds., John Wiley and Sons Inc., 359-373.

[DFJ54] G.B. Dantzig, R. Fulkerson, S. Johnson, (1954). *Solution of a large-scale traveling-salesman problem*. Operational Research 2, 393-410.

[DPST03] J. Dréo, A. Pétrowski, P. Siarry, E. Taillard, (2003). *Métaheuristiques pour l'optimisation difficile*. Eyrolles.

[EFP99] M.L. Espinouse, P. Formanowicz, B. Penz, (1999). *Minimizing the makespan in the two-machine no-wait flow-shop*. Computers and Industrial Engineering, 37, 497-500.

[EFP01] M.L. Espinouse, P. Formanowicz, B. Penz, (1999). *Complexity results and approximation algorithms for the two machine no-wait flow-shop with limited machine availability*. Journal of the Operational Research Society, 52, 1, 116-121.

[EL99] P. Esquirol, P. Lopez, (1999). *Lordonnancement*. Economica, Paris, France.

[EM85] J. Erscher, C. Merce, (1985). *Consistency of the disaggregation process in hierarchical planning*. Operation Research, 34, 464-469.

[FLLRK83] M.L. Fisher, B.J. Lageweg, J.K. Lenstra, A.H.G. Rinnooy Kan, (1983). *Surrogate duality relaxation for job-shop scheduling*. Discrete Applied Mathematics, vol. 5.

[Fre82] S. French, (1982). *Sequencing and Scheduling: An Introduction to the Mathematics of the Job-Shop*. Horwood, Chichester.

[Gaw07] S. Gawiejnowicz, (2007). *Scheduling deteriorating jobs subject to job or machine availability Constraints*. European Journal of Operational Research, 180, 472-478.

[GGS06] J. Gao, M. Gen, L. Sun, (2006). *Scheduling jobs and maintenances in flexible job shop with a hybrid genetic algorithm*. J Intell Manuf, 17, 493-507.

[GH05] A. Gharbi, M. Haouari, (2005). *Optimal parallel machines scheduling with availability constraints*. Discrete Applied Mathematics, 148, 63-87.

[GJ79] M.R. Garey, M.R. Johnson, (1979). *Computers and intractability: A guide to the theory of NP-completeness*. San Francisco: Freeman.

[GL99] G.H. Graves, C.Y. Lee, (1999). *Scheduling maintenance and semiresumable jobs on a single machine*. Naval Research Logistic, 46, 845-863.

[GLLRK79] R.L. Graham, E.L. Lawler, J.K. Lenstra, A.H. G. Rinnooy Kan, (1979). *Optimization and approximation in deterministic sequencing and scheduling: a survey*. Annals Discrete Math., 5, 287 - 326.

[Glo77] F. Glover, (1977). *Heuristics for Integer Programming Using Surrogate Constraints*. Decision Sciences, 8, 1, 156-166.

[Glo86] F. Glover, (1986). *Future Paths for Integer Programming and Links to Artificial Intelligence*. Computers and Operations Research, 13, 533-549.

[GM84] M. Gondran, M. Minoux, (1984), *Graphs and Algorithms*. John Wiley and Sons, New York. 21.

[Gol89] D.E. Goldberg, (1989). *Genetic Algorithms in Search: Optimisation and Machine Learning*. Addison-Wesley, Reading, Massachusetts.

[GTK94] M. Gen, T. Tsujimura, E. Kubota. (1994). *Solving job-shop problem using genetic algorithm*. In proceeding of the 16th International Conference on computers and Industrial Engineering, M.Gen and S.Kobayashi, editors, 576-579. Ashikaga, Japan.

[Har03] Y. Harrath, (2003). *Contribution à l'ordonnancement conjoint de la production et de la maintenance : application au cas d'un Job Shop*. Thèse de PhD, Université de Franche Comté, France.

[HG03] M. Haouari, A. Gharbi, (2003). *An improved max-flow based lower bound for minimizing maximum lateness on identical parallel machines*. Operations Research Letters, 31, 4952.

[Hol75] J. Holland, (1975). *Adaptive in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor.

[Jac55] J.R. Jackson, (1955). *Scheduling a Production Line to Minimise Maximum Tardiness.* Research Report 43, Management Science Research Projects, University of California, Los Angeles, USA.

[JRCW08] J. Jungwattanakit, M. Reodecha, P. Chaowalitwongse, F. Werner, (2008), *Algorithms for Flexible Flow Shop Problems with Unrelated Parallel Machines, Setup Times, and Dual Criteria.* International Journal of Advanced Manufacturing Technology (in press); DOI 10.1007/s00170-007-0977-0.

[Kaa04] J. Kaabi, (2004). *Ordonnancement multicritère des job-shops flexibles : formulation, bornes inférieures et approche évolutionniste coopérative.* Thèse de PhD, Ecole Centrale de Lille, Lille, France.

[Kar84] N. Karmarkar, (1984). *A new polynomial time algorithm for linear programming.* Combinatorica, 4, 373-395.

[Kar72] R.M. Karp, (1972). *Reducibility Among Combinatorial Problems.* in Miller, R.E. and Thatcher, J. W. (eds).Complexity of Computer Computations, Plenum Press, New York, 85-104.

[KBFBS02] W. Kubiak, J. Blazewicz, P. Formanowicz, J. Breit, G. Schmidt, (2002). *Two-machine flow shops with limited machine availability.* European Journal of Operational Research, 136, 528-540.

[KC08a] I. Kacem, C. Chu, (2008). *Efficient branch-and-bound algorithm for minimizing the weighted sum of completion times on a single machine with one availability constraint.* International Journal of Production Economics, 112, 1, 138-150.

[KC08b] I. Kacem, C. Chu, (2008). *Minimizing the weighted flow time on a single machine with the resumable availability constraint: worst case of the WSPT heuristic.* International Journal of Computer Integrated Manufacturing , 21, 4, 388-395.

[KCS08] I. Kacem, C. Chu, A. Souissi, (2008). *Single-machine scheduling with an availability constraint to minimize the weighted sum of the completion times.* Computers and Operations Research, 35, 3, 827-844.

[KGV83] S. Kirkpatrick, C.D. Gellatt, M.P. Vecchi, (1983). *Optimization by Simulated Annealing.* Science, 220, 671-680.

[Kha79] L.G. Khachiyan, (1979). *A polynomial algorithm in linear programming.* Soviet Math. Doklady, 20, 191-194.

[KL93] F. Kraemer, C.Y. Lee, (1993). *Common Due-Window Scheduling.* Production and Operations Management, 2, 262-275.

[KM88] M. Kaspi, B. Montreuil, (1988). *On the scheduling of identical parallel processes with arbitrary initial processor available time.* Research Report 88-12, School of Industrial Engineering, Purdue University.

[KPS09] M.A. Kubzin, C.N. Potts, V.A. Strusevich, (2009), *Approximation results for flow shop scheduling problems with machine availability constraints*. Computers and Operations Research, 36, 2, 379-390.

[KS05] M.A. Kubzin, V.A. Strusevich, (2005). *Two-machine flow shop no-wait scheduling with machine maintenance*. 4OR: A Quarterly Journal of Operations Research, 3, 4, 303-313.

[KS06] M.A. Kubzin, V.A. Strusevich, (2006). *Planning Machine Maintenance in Two-Machine Shop Scheduling*. Operations Research, 54, 4, 789-800.

[KSBS05] M.A. Kubzin, V.A. Strusevich, J. Breit, G. Schmidt, (2005), *Polynomial-time approximation schemes for two-machine open shop scheduling with nonavailability constraints*. Naval Research Logistics, 53, 1, 16-23.

[LBB02a] T. Lorigeon, J.C. Billaut, J.L. Bouquard, (2002), *Availability Constraint for a Single Machine Problem with Heads and Tails*. Project Management and Scheduling, Valence, Spain.

[LBB02b] T. Lorigeon, J.C. Billaut, J.L. Bouquard, (2002), *A dynamic programming algorithm for scheduling jobs in a two-machine open shop with an availability constraint*. Journal of the Operational Research Society, 53, 11, 1239-1246.

[LC95] W. Li, J. Cao, (1995). *Stochastic scheduling on a single machine subject to multiple breakdowns according to different probabilities*. Operations Research Letters, 18, 81-91.

[Lee91] C.Y. Lee, (1991). *Parallel machine scheduling with nonsimultaneous machine available time*. Discrete Applied Mathematics, 30, 53-61.

[Lee96] C.Y. Lee, (1996). *Machine scheduling with an availability constraint*. Journal of Global Optimization, 9, 395-416.

[Lee97] C.Y. Lee, (1997). *Minimizing the makespan in two-machine flowshop scheduling problem with an availability constraint*. Operations Research Letters, 20, 129-139.

[Lee99] C.Y. Lee, (1999). *Two-machine flowshop scheduling with availability constraints*. European Journal of Operational Research, 114, 420-429.

[Leu04] J.Y.T. Leung, (2004), *Handbook of Scheduling*. Chapman & Hall!CRC.

[Lev00] G. Levitin (2000), *Multistate Series-Parallel System Expansion-Scheduling Subject to Availability Constraints*. IEEE Transactions on Reliability, 49, 1, 71-79.

[Lim91] S. Liman, (1991). *Scheduling with Capacities and Due-Dates*. Ph.D. Dissertation, Industrial and Systems Engineering Department, University of Florida.

[LL92] C.Y. Lee, S.D. Liman, (1992). *Single machine flow-time scheduling with scheduled maintenance*. Acta Informatica, 29, 375-382.

[LL93] C.Y. Lee, S.D. Liman, (1993). *Capacitated two-parallel machines scheduling to minimize sum of job completion times.* Discrete Applied Mathematics, 41, 211-222.

[LP93] L. Lu, M.E. Posner, (1993). *An NP-hard open shop scheduling problem with polynomial average time complexity.* Mathematics of Operations Research, 18, 12-38.

[LM89] E.L. Lawler, C.U. Martel, (1989), *Preemptive scheduling of two uniform machines to minimize the number of late jobs.* Operations Research, 37, 314-318.

[LR01] P. Lopez, F. Roubellat, (2001). *Ordonnancement de la Production.* Hermes Science, France.

[LS95a] Z. Liu, E. Sanlaville, (1995). *Preemptive scheduling with variable profile, precedence constraints and due dates.* Discrete Applied Mathematics, 58, 253-280.

[LS97] Z. Liu, E. Sanlaville, (1997). *Stochastic scheduling with variable profile and precedence constraints.* SIAM Journal on Computing, 26, 173-187.

[LW91] L. Lei, T. J. Wong, (1991). *The Minimum Common-Cycle Algorithm for Cycling Scheduling of Two Material Handling Hoists with Time Window Constraints.* Management Science, 37, 1629-1639.

[LW92] V.J. Leon, S.D. Wu, (1992). *On scheduling with ready-times, due-dates and vacations.* Naval Research Logistics, 39, 53-65.

[LYH98] G.H. Lin, E.Y. Yao, Y. He, (1998). *Parallel machine scheduling to maximize the minimum load with non simultaneous machine available times.* Operations Research Letters, 22, 75-81.

[MBB03a] P. Mauguière, J. C. Billaut, J. L. Bouquard, (2003a), *Scheduling resumable and non-resumable Operations.* In: Proceedings of the Joint International Meeting EURO/INFORMS, Istanbul (Turkey), 166-167.

[MBB03b] P. Mauguière, J. L. Bouquard, J. C. Billaut, (2003b), *A branch and bound algorithm for a job shop scheduling problem with availability constraints.* In: Proceedings of the Sixth Workshop on Models and Algorithms for Planning and Scheduling Problems, MAPSP'2003, Aussois (France), 147-148.

[MBB05] Ph. Mauguière, J.C. Billaut, J.L. Bouquard, (2005). *New single machine and job-shop scheduling problems with availability constraints.* Journal of Scheduling, 8, 3, 211-231.

[MF75] G.B. Mac Maron, M. Florian, (1975). *On Scheduling with Ready Time and Due Dates to Minimize Maximum Lateness.* Operations Research, 23, 3, 475-482.

[Mos94] G. Mosheiov, (1994). *Minimizing the sum of job completion times on capacitated parallel machines.* Mathl. Comput. Modelling, 20, 91-99.

[MSCK09] R. Mellouli, C. Sadfi, C. Chu, I. Kacem, (2009). *Identical parallel-machine scheduling under availability constraints to minimize the sum of completion times.* European Journal of Operational Research, 197, 3, 1150-1165.

[NK03] C.T. Ng, M.Y. Kovalyov, (2003). *An FPTAS for scheduling a two-machine flowshop with one unavailability interval.* Naval Research Logistics, 51, 3, 307-315.

[Pin95] M. Pinedo, (1995). *Scheduling : Theory, Algorithms, and Systems.* Prentice-Hall, Englewood Cliffs, New Jersey, Etats-Unis.

[Por88] M.C. Portmann, (1988). *Méthodes de décomposition spatiales et temporelles en ordonnancement.* RAIRO-APII, 22, 5, 439-451.

[PWW69] A.A.B. Pritsker, L.J. Watters, P.M. Wolfe, (1969). *Multi-Project Scheduling with Limited Resources: A Zero-One Programming Approach.* Management Science 16.

[QCT99] T. Qi, T. Chen, F. Tu, (1999). *Scheduling the maintenance on a single machine.* Journal of Operational Research Society, 50, 1071-1078.

[Ree95] C.R. Reeves, (1995). *A genetic algorithm for flowshop sequencing.* Computers and Research, 22, 5-14.

[RK76] A.H.G. Rinnooy Kan, (1976). *Machine Scheduling Problems: Classification, Complexity and Computations.* Martinus Nijhoff, The Haque.

[RS64] B. Roy, B. Susmann, (1964). *Les Problèmes dordonnancement avec Contraintes Disjonctives.* Note DS n 9 bis, SEMA, Montrouge.

[Sad02] C. Sadfi, (2002). *Problèmes d'ordonnancement avec minimisation des encours.* Thèse de PhD, GILCO, INP Grenoble, France.

[San95] E. Sanlaville, (1995), *Nearly on line scheduling of preemptive independent tasks.* Discrete Applied Mathematics, 57, 229-241.

[Sch84] G. Schmidt, (1984). *Scheduling on semi-identical processors.* Z. Oper. Res., A28, 153-162.

[Sch88] G. Schmidt, (1988). *Scheduling independent tasks with deadlines on semi-identical processors.* Journal of Operational Research Society, 39, 271-277.

[SL07] G. J. Sheen, L. W. Liao, (2007), *Scheduling machine-dependent jobs to minimize lateness on machines with identical speed under availability constraints.* Computers and Operations Research, 34, 8, 2266-2278.

[Sou05] A.S. Souissi, (2005). *Ordonnancement avec prise en compte des indisponibilités dépendantes et indépendantes.* Thèse de PhD. Université de Technologie de Troyes, France.

[SPRBF05] C. Sadfi, B. Penz, C. Rapine, J. Blazewicz, P. Formanowicz, (2005). *An improved approximation algorithm for the single machine total completion time scheduling problem with availability constraints: IEPM: Focus on scheduling.* European journal of operational research, 161, 1, 3-10.

[SS98] E. Sanlaville, G. Schmidt, (1998). *Machine scheduling with availability constraints.* Acta Informatica, 35, 795-811.

[Tal09] E.G. Talbi, (2009), *Metaheuristics From Design to Implementation.* Wiley.

[TFDS09] M.T. Taghavi-Fard, H.R. Dehnar Saidy, (2009), *Flexible Job Shop Scheduling Under Availability Constraints.* Journal of Industrial Engineering International, 5, 8, 52-60.

[TGGP01] E.D. Taillard, L.M. Gambardella, M. Gendreau, J.Y. Potvin, (2001). *Adaptive Memory Programming: A Unified View of Metaheuristics.* European Journal of Operational Research, 135, 1, 1-16.

[TGS94a] V.S. Tanaev, V.S. Gordon, Y.M. Shafransky, (1994). *Scheduling theory. Single-stage systems.* Kluwer Academic Publishers. Dordrecht / Boston / London.

[Ull75] J.D. Ullman, (1975). *NP-complete scheduling problems.* Journal of Computer and System Sciences, 10, 384-393.

[VS95] G. Vairaktarakis, S. Sahni, (1995). *Dual criteria preemptive open-shop problems with minimum makespan.* Naval Research Logistics, 42, 103-121.

[WC01] G. Wang, T.C.E. Cheng, (2001). *Heuristics for two-machine no-wait flowshop scheduling with an availability constraint.* Information Processing Letters, 80, 6, 305-309.

[WC06] X. Wang, T.C.E. Cheng, (2006), *Machine scheduling with an availability constraint and job delivery Coordination.* Naval Research Logistics, 54, 1, 11-20.

[WC07a] X. Wang, T.C.E. Cheng, (2007), *Heuristics for two-machine flowshop scheduling with setup times and an availability constraint.* Computers and Operations Research 34, 152-162.

[WC07b] X. Wang, T.C.E. Cheng, (2007), *An approximation scheme for two-machine flowshop scheduling with setup times and an availability constraint.* Computers and Operations Research, 34, 10, 2894-2901.

[WL03] C.C. Wu, W.C. Lee, (2003). *Scheduling linear deteriorating jobs to minimize makespan with an availability constraint on a single machine.* Information Processing Letters, 87, 89-93.

[Zri05] N. Zribi, (2005). *Ordonnancement des job-shops flexibles sous contraintes de disponibilité des machines.* Thèse de PhD, Université des Sciences et Technologies de Lille, Ecole Centrale de Lille, Lille, France.

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : **2010 EMSE 0574**

Sadia AZEM

Dissertation title
Scheduling flexible production systems under resource availability constraints

Major
Industrial Engineering

Keywords
Scheduling, flexibility, optimization, availability constraints, industrial systems, software programming, mathematical modeling, approximate methods, column generation

Abstract

In most of the machine scheduling literature, resources are assumed to be continuously available which is not always true. We deal with the context of unavailability known a priori; we are particularly interested in job-shop scheduling problems with flexible unavailability periods and tasks that can eventually be interrupted by unavailability periods. Integrating these constraints increase the complexity of the scheduling problems. We deal with flexibility that is related to at least one of the following points: moving the unavailability period in a time window, modification of the duration of the unavailability period, interruption of a task by an unavailability period, then resumed with a possible penalty.

In this thesis, we propose mathematical models for the problem. In addition to the resolution of the considered problems, the aim of this modeling is to allow for the analysis of the impact of different constraints and evaluation of the quality of the approximate methods and the column generation approach we develop. The approximate methods construct in very short time a schedule based on priority rules. The solutions are also used in our column generation approach. This approach adapts well to various objective functions and allows relatively easily for the integration of several constraints. Many experiments have been performed to validate the designed methods.

École Nationale Supérieure des Mines
de Saint-Étienne

N° d'ordre : **2010 EMSE 0574**

Sadia AZEM

Titre de la thèse
Ordonnancement des systèmes flexibles de production sous contraintes de disponibilité des ressources

Spécialité
Génie Industriel

Mots clefs
Ordonnancement, flexibilité, optimisation, contraintes de disponibilité, systèmes industriels, génie logiciel, modélisation mathématique, méthodes approchées, génération de colonnes.

Résumé
La majeure partie des travaux sur les problèmes d'ordonnancement se placent dans le contexte où les ressources sont disponibles en permanence. Ce qui en réalité n'est pas toujours le cas. Nous nous plaçons dans le contexte d'indisponibilités connues ; nous nous intéressons plus particulièrement aux problèmes de type job shop avec des périodes d'indisponibilité flexibles et des tâches pouvant éventuellement être interrompues par les périodes d'indisponibilité. L'intégration de ces contraintes rend les problèmes d'ordonnancement nettement plus difficiles à résoudre. La flexibilité que nous considérons peut être relative à au moins l'un des points suivants : déplacement de la période d'indisponibilité dans une fenêtre de temps, modification de la durée de la période d'indisponibilité, interruption d'une tâche par une période d'indisponibilité, ensuite reprise avec une éventuelle pénalité.

Dans cette thèse, nous avons proposé des modèles mathématiques pour le problème. En plus de la résolution des problèmes considérés, le but de ces modélisations est de permettre d'analyser l'impact des différentes contraintes et d'évaluer la qualité des méthodes approchées que nous proposons. Ces dernières permettent de construire très rapidement un ordonnancement en se basant sur des règles de priorité. Les solutions sont aussi utilisées pour notre approche basée sur la génération de colonnes. Cette approche s'adapte bien à différents fonctions objectif et permet d'intégrer relativement facilement plusieurs contraintes. De nombreuses expérimentations ont été menées pour valider les méthodes proposées.